**CONVERT TO NORTHWINDS**

**Code in notebook:**

**Northwind and Tsql**

---------------------------------------------------------------------

-- Microsoft SQL Server T-SQL Fundamentals

-- Chapter 03 - Joins

-- © Itzik Ben-Gan

---------------------------------------------------------------------

---------------------------------------------------------------------

-- CROSS Joins

---------------------------------------------------------------------

USE TSQLV4;

-- SQL-92

SELECT C.custid, E.empid

FROM Sales.Customers AS C

CROSS JOIN HR.Employees AS E;

USE Northwinds2022TSQLV7;

SELECT C.CustomerId, E.EmployeeId

FROM Sales.Customer AS C

CROSS JOIN HumanResources.Employee AS E;

* Retrieve combinations of CustomerId and EmployeeId from Sales.Customer and HumanResources.Employee tables
* Implement a CROSS JOIN to obtain all possible combinations between the two tables
* Tables involved: Sales.Customer and HumanResources.Employee

-- SQL-89

SELECT C.custid, E.empid

FROM Sales.Customers AS C, HR.Employees AS E;

Use Northwinds2022TSQLV7

SELECT C.CustomerId, E.EmployeeId

FROM Sales.Customer as C, HumanResources.Employee as E;

PROPOSITION: Write a query that returns the customerId from the Sales.Customer Table and Employee Id from the HumanResoruces.Employee table.

TABLES: Northwinds2022TSQLV7 Database, Sales.Customer as alias C and HumanResources.Employee as alias E.

COLUMN: Customer from Sales.Customer and EmployeeId from HumanResorces.Employee table.

PREDICATE: While not explicitly defined, there is a LEFT OUTER JOIN between Sales.Customer and HumanResources..Employee ON CustomerId being identical.

-- Self Cross-Join

SELECT

E1.empid, E1.firstname, E1.lastname,

E2.empid, E2.firstname, E2.lastname

FROM HR.Employees AS E1

CROSS JOIN HR.Employees AS E2;

GO

-- All numbers from 1 - 1000

-- Auxiliary table of digits

USE TSQLV4;

DROP TABLE IF EXISTS dbo.Digits;

CREATE TABLE dbo.Digits(digit INT NOT NULL PRIMARY KEY);

INSERT INTO dbo.Digits(digit)

VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9);

SELECT digit FROM dbo.Digits;

GO

USE Northwinds2022TSQLV7;

DROP TABLE IF EXISTS dbo.Digits;

CREATE TABLE dbo.Digits(digit INT NOT NULL PRIMARY KEY);

INSERT INTO dbo.Digits(digit)

VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9);

SELECT digit FROM dbo.Digits;

GO

* Generate a sequence of digits from 0 to 9
* Store these digits in a table named "Digits"
* Provide a simple query to select and display these digits
* Imagine you have a table that stores single-digit numbers from 0 to 9. This table is called "Digits."

-- All numbers from 1 - 1000

SELECT D3.digit \* 100 + D2.digit \* 10 + D1.digit + 1 AS n

FROM dbo.Digits AS D1

CROSS JOIN dbo.Digits AS D2

CROSS JOIN dbo.Digits AS D3

ORDER BY n;

---------------------------------------------------------------------

-- INNER Joins

---------------------------------------------------------------------

USE TSQLV4;

-- SQL-92

SELECT E.empid, E.firstname, E.lastname, O.orderid

FROM HR.Employees AS E

INNER JOIN Sales.Orders AS O

ON E.empid = O.empid;

USE Northwinds2022TSQLV7;

SELECT E.EmployeeId, E.EmployeeFirstName, E.EmployeeLastName, O.OrderId

FROM HumanResources.Employee AS E

INNER JOIN Sales.[Order] AS O

ON E.EmployeeId = O.EmployeeId;

-Proposition: return employee id, employee's name, and the order id

-table involved: HumanResources.Employee, Sales.[Order]

-Columns: EmployeeId, EmployeeFirstName, EmployeeLastName from HumanResources.Employee. OrderId from Sales.[Order]

-- SQL-89

SELECT E.empid, E.firstname, E.lastname, O.orderid

FROM HR.Employees AS E, Sales.Orders AS O

WHERE E.empid = O.empid;

GO

-- Inner Join Safety

/\*

SELECT E.empid, E.firstname, E.lastname, O.orderid

FROM HR.Employees AS E

INNER JOIN Sales.Orders AS O;

GO

\*/

/\*USE Northwinds2022TSQLV7;

SELECT E.EmployeeId, E.EmployeeFirstName, E.EmployeeLastName, O.OrderId

FROM HumanResources.Employee AS E

INNER JOIN Sales.[Order] AS O;

GO

\*/

* Retrieve employee information and associated order details
* Match employees with their corresponding orders
* Utilize tables: HumanResources.Employee (referred to as E) and Sales.Order (referred to as O)
* Picture a scenario where we want to gather information about employees and the orders they are associated with. This involves bringing together data from two tables: one containing employee details (like first name, last name, and ID) called "Employee," and another holding order information (such as order ID) named "Order."

SELECT E.empid, E.firstname, E.lastname, O.orderid

FROM HR.Employees AS E, Sales.Orders AS O;

GO

Use Northwinds2022TSQLV7

SELECT E.EmployeeId, E.EmployeeFirstName, E.EmployeeLastName, O.orderid

FROM HumanResources.Employee AS E, Sales.[Order] AS O;

GO

PROPOSITION: Write a query that returns EmployeeId, EmployeeFirstName, EmployeeLastName from the HumanResources.Employee table and OrderId from the Sales.[Order] table, resulting in a Cartesian product of all combinations of rows between the two tables (cross join).

TABLES: Northwinds2022TSQLV7 Database, HumanResources.Employee as alias E and Sales.[Order] as alias O.

COLUMN: EmployeeId, EmployeeFirstName and EmployeeLastName from HumanResources.Employee table and orderid from Sales.[Order] table.

PREDICATE: No explicit 'WHERE' or 'JOIN' conditions, however the predicate is the combination of all possible pairs of rows between the HumanResources.Employee and Sales.[Order] tables, which results in a Cartesian product.

---------------------------------------------------------------------

-- More Join Examples

---------------------------------------------------------------------

---------------------------------------------------------------------

-- Composite Joins

---------------------------------------------------------------------

-- Audit table for updates against OrderDetails

USE TSQLV4;

DROP TABLE IF EXISTS Sales.OrderDetailsAudit;

CREATE TABLE Sales.OrderDetailsAudit

(

lsn INT NOT NULL IDENTITY,

orderid INT NOT NULL,

productid INT NOT NULL,

dt DATETIME NOT NULL,

loginname sysname NOT NULL,

columnname sysname NOT NULL,

oldval SQL\_VARIANT,

newval SQL\_VARIANT,

CONSTRAINT PK\_OrderDetailsAudit PRIMARY KEY(lsn),

CONSTRAINT FK\_OrderDetailsAudit\_OrderDetails

FOREIGN KEY(orderid, productid)

REFERENCES Sales.OrderDetails(orderid, productid)

);

SELECT OD.orderid, OD.productid, OD.qty,

ODA.dt, ODA.loginname, ODA.oldval, ODA.newval

FROM Sales.OrderDetails AS OD

INNER JOIN Sales.OrderDetailsAudit AS ODA

ON OD.orderid = ODA.orderid

AND OD.productid = ODA.productid

WHERE ODA.columnname = N'qty';

---------------------------------------------------------------------

-- Non-Equi Joins

---------------------------------------------------------------------

-- Unique pairs of employees

SELECT

E1.empid, E1.firstname, E1.lastname,

E2.empid, E2.firstname, E2.lastname

FROM HR.Employees AS E1

INNER JOIN HR.Employees AS E2

ON E1.empid < E2.empid;

USE Northwinds2022TSQLV7;

SELECT

E1.EmployeeId, E1.EmployeeFirstName, E1.EmployeeLastName,

E2.EmployeeId, E2.EmployeeFirstName, E2.EmployeeLastName

FROM HumanResources.Employee AS E1

INNER JOIN HumanResources.Employee AS E2

ON E1.EmployeeId < E2.EmployeeId;

* Establish relationships between pairs of employees
* Select pairs with the first employee having a lower ID than the second
* Utilize the HumanResources.Employee table (referred to as E1 and E2)
* Imagine you want to create pairs of employees, showcasing relationships between them. This query involves using information from the Employee table twice, labeling them as E1 and E2.

---------------------------------------------------------------------

-- Multi-Join Queries

---------------------------------------------------------------------

SELECT

C.custid, C.companyname, O.orderid,

OD.productid, OD.qty

FROM Sales.Customers AS C

INNER JOIN Sales.Orders AS O

ON C.custid = O.custid

INNER JOIN Sales.OrderDetails AS OD

ON O.orderid = OD.orderid;

Use Northwinds2022TSQLV7

SELECT C.CustomerId, C.CustomerCompanyName, O.orderid, OD.productid, OD.Quantity

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

ON C.CustomerId = O.CustomerId

INNER JOIN Sales.OrderDetail AS OD

ON O.orderid = OD.orderid;

PROPOSITION: Write a query that returns the CustomerId and CustomerCompanyName from the Sales.Customer table, OrderId from the Sales.[Order] table, ProductId, and Quantity from the Sales.OrderDetail table. The Query should Join Sales.Customer with Sales.[Order] on CustomerId, and then joins Sales.[Order] with Sales.OrderDetail on orderid.

TABLES: Northwinds2022TSQLV7 Database, Sales.Customer as alias C, Sales.[Order] as alias O and Sales.OrderDetails as alias OD

COLUMN: CustomerId and CustomerCompanyName from the Sales.Customer table, OrderId from the Sales.[Order] table, ProductId, and Quantity from the Sales.OrderDetail table

PREDICATE: INNER JOIN Sales.[Order] and Sales.Customer ON CustomerId being equal and INNER JOIN Sales.OrderDetail and Sales.Customer ON orderId being equal.

---------------------------------------------------------------------

-- Fundamentals of Outer Joins

---------------------------------------------------------------------

-- Customers and their orders, including customers with no orders

SELECT C.custid, C.companyname, O.orderid

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid;

-- Customers with no orders

SELECT C.custid, C.companyname

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid

WHERE O.orderid IS NULL;

USE Northwinds2022TSQLV7

SELECT C.CustomerId, C.CustomerCompanyName

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O

ON C.CustomerId = O.CustomerId

WHERE O.orderid IS NULL;

-Proposition: return customer id and company that has not placed any order

-table involved: Sales.Customer, Sales.[Order]

-Columns: CustomerId, CustomerCompanyName from Sales.Customer

-Predicate: the where clause O.orderid IS NULL filters out all customer id and company name with no order

---------------------------------------------------------------------

-- Beyond the Fundamentals of Outer Joins

---------------------------------------------------------------------

---------------------------------------------------------------------

-- Including Missing Values

---------------------------------------------------------------------

SELECT DATEADD(day, n-1, CAST('20140101' AS DATE)) AS orderdate

FROM dbo.Nums

WHERE n <= DATEDIFF(day, '20140101', '20161231') + 1

ORDER BY orderdate;

SELECT DATEADD(day, Nums.n - 1, CAST('20140101' AS DATE)) AS orderdate,

O.orderid, O.custid, O.empid

FROM dbo.Nums

LEFT OUTER JOIN Sales.Orders AS O

ON DATEADD(day, Nums.n - 1, CAST('20140101' AS DATE)) = O.orderdate

WHERE Nums.n <= DATEDIFF(day, '20140101', '20161231') + 1

ORDER BY orderdate;

USE Northwinds2022TSQLV7;

SELECT DATEADD(day, Nums.n - 1, CAST('20140101' AS DATE)) AS orderdate,

O.OrderId, O.CustomerId, O.EmployeeId

FROM dbo.Nums

LEFT OUTER JOIN Sales.[Order] AS O

ON DATEADD(day, Nums.n - 1, CAST('20140101' AS DATE)) = O.orderdate

WHERE Nums.n <= DATEDIFF(day, '20140101', '20161231') + 1

ORDER BY orderdate;

* Generate a sequence of dates starting from '20140101'
* Match each date with corresponding order details
* Utilize a table called "Nums" to create the date sequence
* Fetch order information like Order ID, Customer ID, and Employee ID from the Sales.Order table
* Imagine you want to create a list of dates and find the associated orders for each date. This involves generating a sequence of dates starting from '20140101' and connecting them with order details from the Sales.Order table.

---------------------------------------------------------------------

-- Filtering Attributes from Non-Preserved Side of Outer Join

---------------------------------------------------------------------

SELECT C.custid, C.companyname, O.orderid, O.orderdate

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid

WHERE O.orderdate >= '20160101';

Use Northwinds2022TSQLV7

SELECT C.CustomerId, C.CustomerCompanyName, O.orderid, O.orderdate

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O

ON C.CustomerId = O.CustomerId

WHERE O.orderdate >= '20160101';

PROPOSITION: Write a query that returns the CustomerId and CustomerCompanyName from the Sales.Customer table, OrderId and OrderDate from the Sales.[Order] table. Also perform a LEFT OUTER JOIN between Sales.Customer and Sales.[Order] on CustomerId, and filter the result set to include only rows where the orderdate is greater than or equal to January 1, 2016.

TABLES: Northwinds2022TSQLV7 Database, Sales.Customer as alias C and Sales.[Order] as alias O.

COLUMN: CustomerId and CustomerCompanyName from the Sales.Customer table and orderId and orderdate from the Sales.[Order] table.

PREDICATE: LEFT OUTER JOIN between Sales.Customer table and Sales.[Order] table WHERE CustomerId in both tables must be equal and orderdate from Sales.[Order] WHERE orderdate >= to January 1, 2016.

---------------------------------------------------------------------

-- Using Outer Joins in a Multi-Join Query

---------------------------------------------------------------------

SELECT C.custid, O.orderid, OD.productid, OD.qty

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid

INNER JOIN Sales.OrderDetails AS OD

ON O.orderid = OD.orderid;

-- Option 1: use outer join all along

SELECT C.custid, O.orderid, OD.productid, OD.qty

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid

LEFT OUTER JOIN Sales.OrderDetails AS OD

ON O.orderid = OD.orderid;

USE Northwinds2022TSQLV7

SELECT C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O

ON C.CustomerId = O.CustomerId

LEFT OUTER JOIN Sales.OrderDetail AS OD

ON O.OrderId = OD.OrderId;

-Proposition: return customer id, order id, product id and quantity

-table involved: Sales.Customer, Sales.[Order], Sales.OrderDetail

-Columns: CustomerId from Sales.Customer, OrderId from Sales.[Order], ProductId, Quantity from Sales.OrderDetail

-- Option 2: change join order

SELECT C.custid, O.orderid, OD.productid, OD.qty

FROM Sales.Orders AS O

INNER JOIN Sales.OrderDetails AS OD

ON O.orderid = OD.orderid

RIGHT OUTER JOIN Sales.Customers AS C

ON O.custid = C.custid;

-- Option 3: use parentheses

SELECT C.custid, O.orderid, OD.productid, OD.qty

FROM Sales.Customers AS C

LEFT OUTER JOIN

(Sales.Orders AS O

INNER JOIN Sales.OrderDetails AS OD

ON O.orderid = OD.orderid)

ON C.custid = O.custid;

USE Northwinds2022TSQLV7;

SELECT C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity

FROM Sales.Customer AS C

LEFT OUTER JOIN

(Sales.[Order] AS O

INNER JOIN Sales.OrderDetail AS OD

ON O.orderid = OD.orderid)

ON C.CustomerId = O.CustomerId;

* Retrieve information about customers, their orders, and associated order details
* Connect customer data with order data, including product details and quantities
* Utilize tables: Sales.Customer (referred to as C), Sales.Order (referred to as O), and Sales.OrderDetail (referred to as OD)
* Imagine you want to gather comprehensive information about customers, their orders, and the products they've purchased. This query involves connecting data from three tables: Customer, Order, and OrderDetail.

---------------------------------------------------------------------

-- Using the COUNT Aggregate with Outer Joins

---------------------------------------------------------------------

SELECT C.custid, COUNT(\*) AS numorders

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid

GROUP BY C.custid;

Use Northwinds2022TSQLV7

SELECT C.CustomerId, COUNT(\*) AS numorders

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O

ON C.CustomerId = O.CustomerId

GROUP BY C.CustomerId;

PROPOSITION: Write a query that returns the CustomerId from the Sales.Customer table and counts the number of orders for each customer using a LEFT OUTER JOIN with the Sales.[Order] table on the CustomerId column. The result is grouped by CustomerId.

TABLES: Northwinds2022TSQLV7 Database, Sales.Customer as alias C and Sales.[Order] as alias O.

COLUMN: CustomerId from the Sales.Customer table CustomerId from the Sales.[Order] table.

PREDICATE: LEFT OUTER JOIN between Sales.Customer table and Sales.[Order] table ON the CustomerId in both tables are identical.

SELECT C.custid, COUNT(O.orderid) AS numorders

FROM Sales.Customers AS C

LEFT OUTER JOIN Sales.Orders AS O

ON C.custid = O.custid

GROUP BY C.custid;

USE Northwinds2022TSQLV7

SELECT C.CustomerId, COUNT(O.OrderId) AS numorders

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O

ON C.CustomerId = O.CustomerId

GROUP BY C.CustomerId;

-Proposition: return customer id and the number of orders they ordered

-table involved: Sales.Customer, Sales.[Order]

-Columns: CustomerId from Sales.Customer, numorders from COUNT(O.OrderId) from Sales.[Order]