Assignment No: 3(Joins)

Name: Pratik Wani

SQL Queries

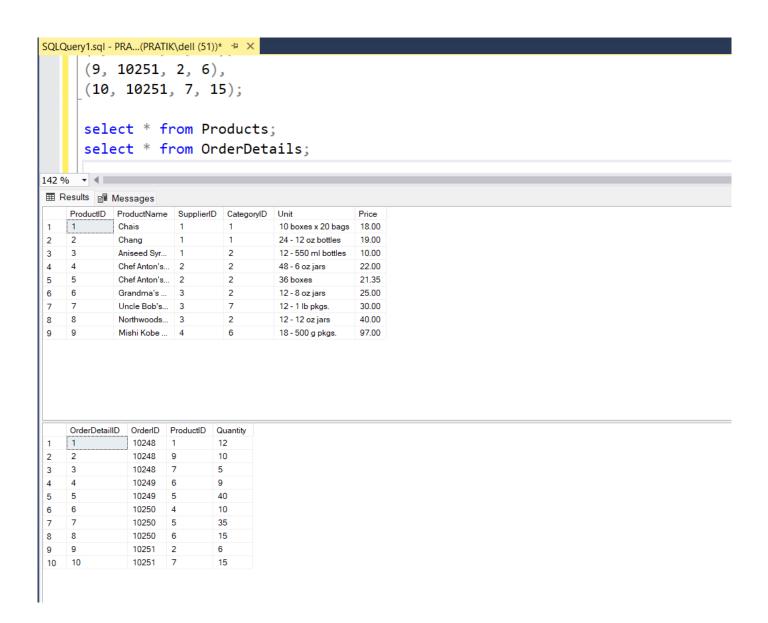
- Here I created a database named as 'Practice' and table named as 'Products' and 'OrderDetails' performed all the Join types.
- Create database, Create tables and Inserted Data:
 - Create is a DDL command
 - Creating a database doesn't automatically set it as the active database, so we need USE command
 - While Creating Table we need to give all the column names there data types and constraints

```
SQLQuery1.sql - PRA...(PRATIK\dell (51))* 😕 🔀
 ----Assignment 3
 use practice;
 -- Create the Products table
□CREATE TABLE Products (
     ProductID INT PRIMARY KEY,
     ProductName NVARCHAR(255),
     SupplierID INT,
     CategoryID INT,
     Unit NVARCHAR(50),
     Price DECIMAL(10, 2)
 );
 -- Insert data into the Products table
垣 INSERT INTO Products (ProductID, ProductName, SupplierID, CategoryID, Unit, Price)
 VALUES
 (1, 'Chais', 1, 1, '10 boxes x 20 bags', 18),
 (2, 'Chang', 1, 1, '24 - 12 oz bottles', 19),
 (3, 'Aniseed Syrup', 1, 2, '12 - 550 ml bottles', 10),
 (4, 'Chef Anton''s Cajun Seasoning', 2, 2, '48 - 6 oz jars', 22),
 (5, 'Chef Anton''s Gumbo Mix', 2, 2, '36 boxes', 21.35),
 (6, 'Grandma''s Boysenberry Spread', 3, 2, '12 - 8 oz jars', 25),
 (7, 'Uncle Bob''s Organic Dried Pears', 3, 7, '12 - 1 lb pkgs.', 30),
  (8, 'Northwoods Cranberry Sauce', 3, 2, '12 - 12 oz jars', 40),
  (9, 'Mishi Kobe Niku', 4, 6, '18 - 500 g pkgs.', 97);
```

```
SQLQuery1.sql - PRA...(PRATIK\dell (51))* 垣 🗶
 -- Create the OrderDetails table
OrderDetailID INT PRIMARY KEY,
     OrderID INT,
     ProductID INT references products(productid),
     Quantity INT
 );
 -- Insert data into the OrderDetails table
□INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity)
 VALUES
 (1, 10248, 1, 12),
 (2, 10248, 9, 10),
 (3, 10248, 7, 5),
 (4, 10249, 6, 9),
 (5, 10249, 5, 40),
 (6, 10250, 4, 10),
 (7, 10250, 5, 35),
 (8, 10250, 6, 15),
 (9, 10251, 2, 6),
 (10, 10251, 7, 15);
```

Select:

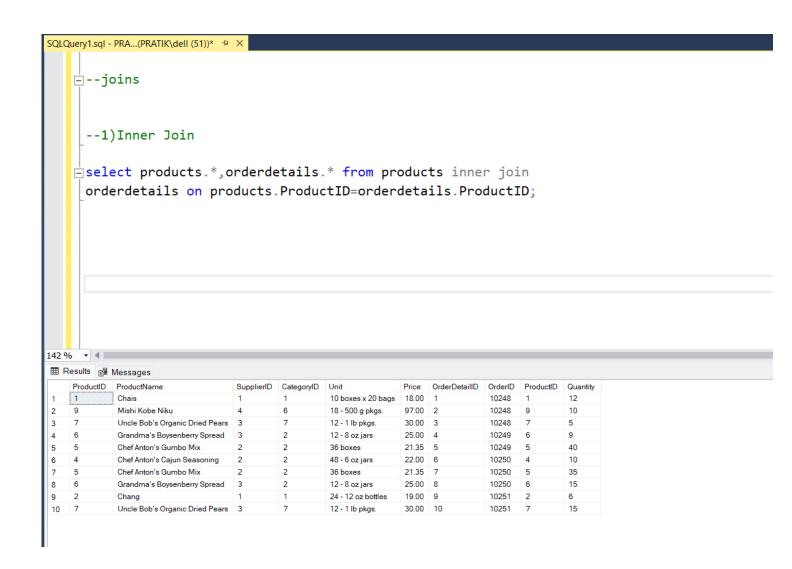
Select * gives all records



Joins:

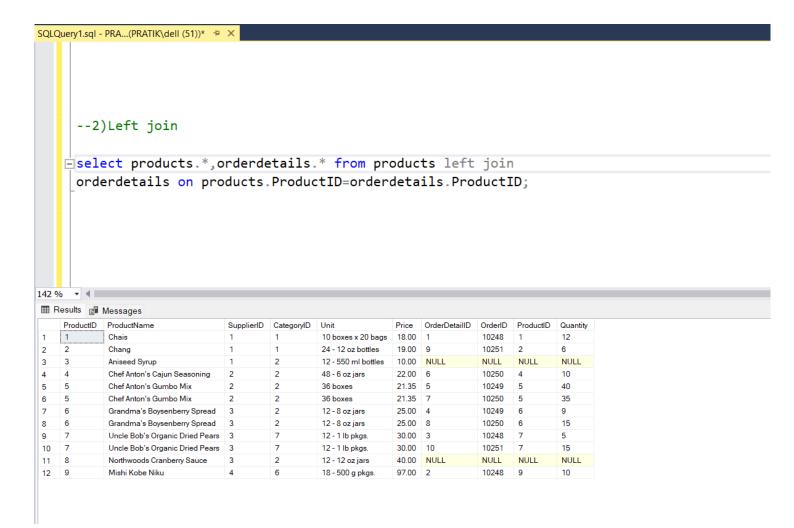
1. Inner Join:

- The INNER JOIN keyword selects all rows from both the tables as long the condition is satisfied.
- This keyword will create the result-set by combining all rows from both the tables where the condition satisfies
- We need some common columns to perform the inner join



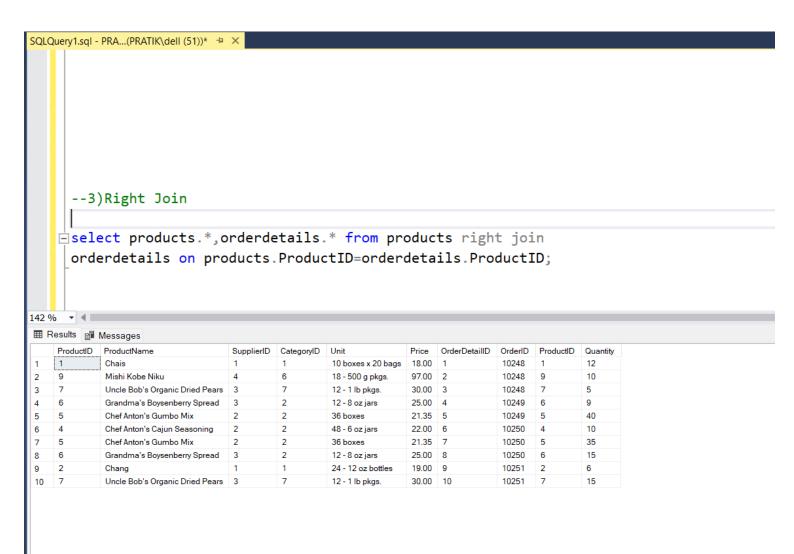
2. Left Join:

- This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join.
- For the rows for which there is no matching row on the right side, the resultset will contain null.
- Also known as Left Outer Join



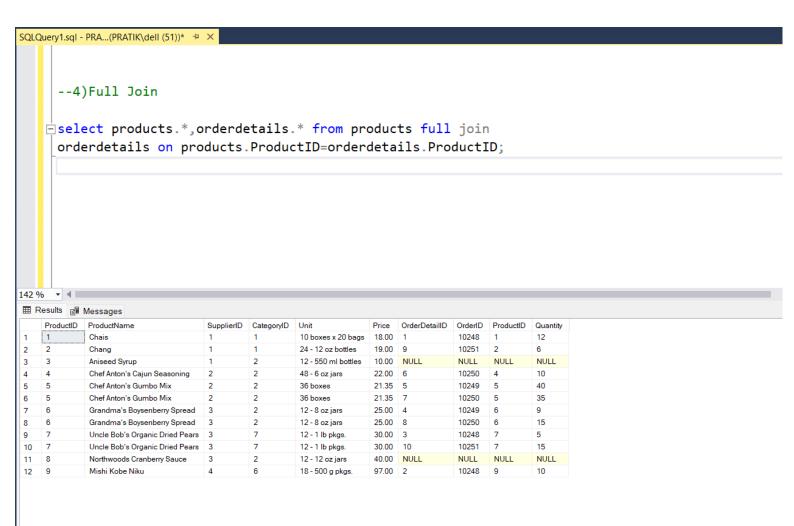
3. Right Join:

- This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join.
- For the rows for which there is no matching row on the left side, the result-set will contain null.
- Also known as Right Outer Join.



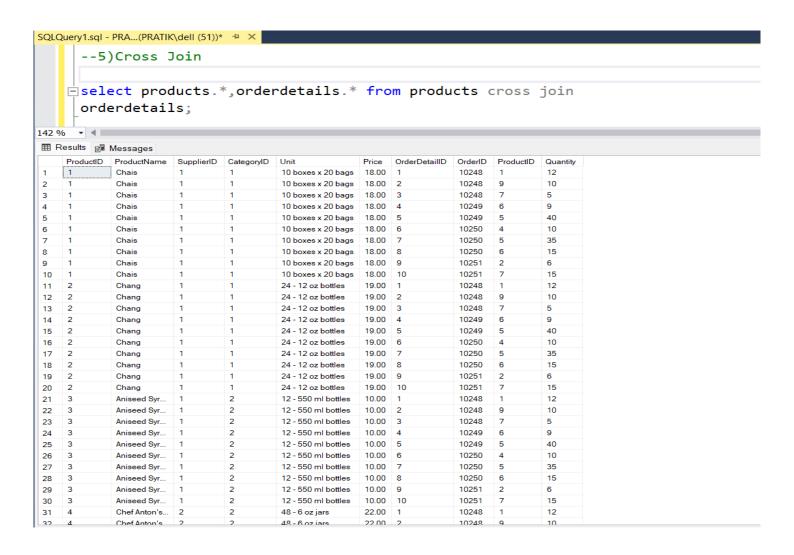
4. Full Join:

- FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN.
- The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.
- Also known as Full Outer Join.



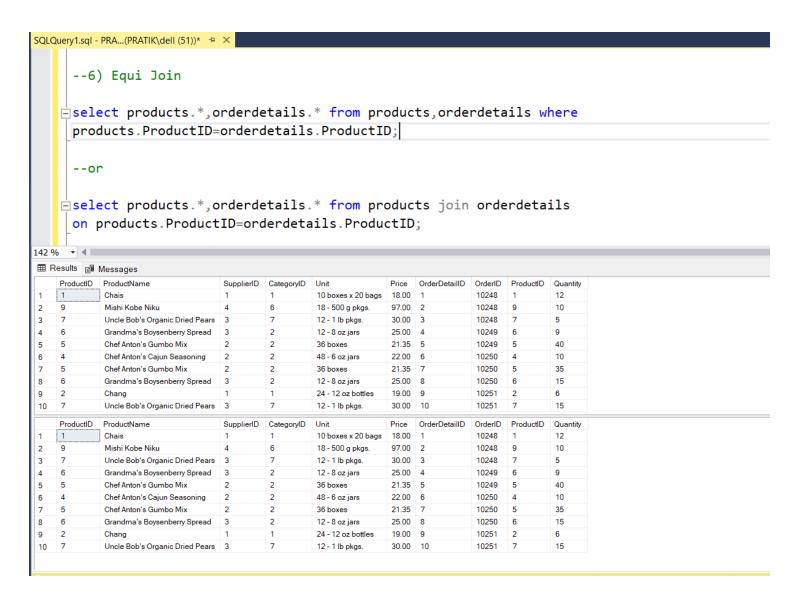
5. Cross join:

- The Cross Join keyword returns all matching records from both tables whether the other table matches or not.
- o Means simply we can say that it's cartesian product of two tables.



6. Equi Join:

- EQUI JOIN creates a JOIN for equality or matching column(s) values of the relative tables.
- EQUI JOIN also create JOIN by using JOIN with ON and then providing the names of the columns with their relative tables to check equality using equal sign (=).



7. Non-Equi Join:

- NON EQUI JOIN performs a JOIN using comparison operator other than equal
 (=) sign like >, <, >=, <= with conditions.
- o Instead of = we use >, <</p>

