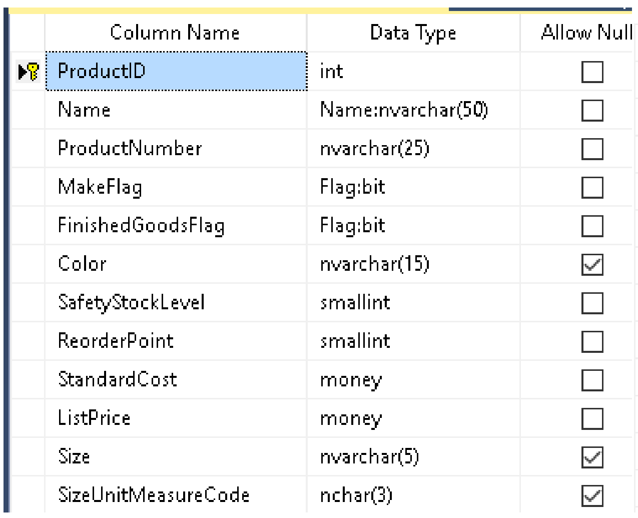
LABORATORY 2

Let us consider the **AdventureWorks2012** database – Full Database Backup (Please Download it from https://msftdbprodsamples.codeplex.com/downloads/get/417885) and Restore it.

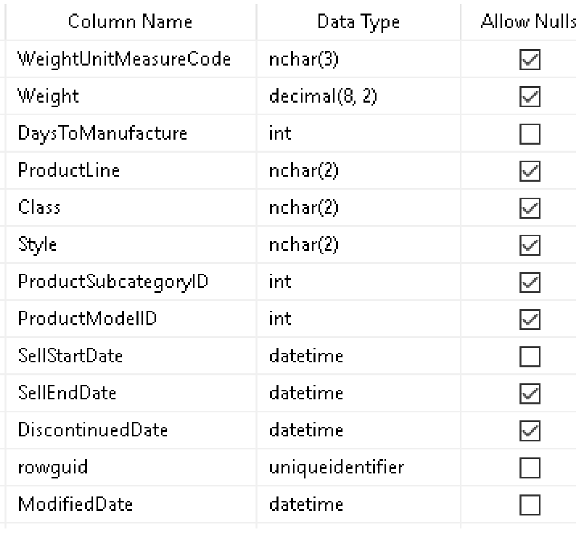
USE AdventureWorks2012
GO

WHERE

Table Production.Product



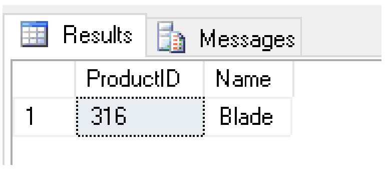
Column Name	Data Type	Allow Null
ProductID	int	☐
Name	Name:nvarchar(50)	☐
ProductNumber	nvarchar(25)	☐
MakeFlag	Flag:bit	☐
FinishedGoodsFlag	Flag:bit	☐
Color	nvarchar(15)	☑
SafetyStockLevel	smallint	☐
ReorderPoint	smallint	☐
StandardCost	money	☐
ListPrice	money	☐
Size	nvarchar(5)	☑
SizeUnitMeasureCode	nchar(3)	☑



Column Name	Data Type	Allow Nulls
WeightUnitMeasureCode	nchar(3)	☑
Weight	decimal(8, 2)	☑
DaysToManufacture	int	☐
ProductLine	nchar(2)	☑
Class	nchar(2)	☑
Style	nchar(2)	☑
ProductSubcategoryID	int	☑
ProductModelID	int	☑
SellStartDate	datetime	☐
SellEndDate	datetime	☑
DiscontinuedDate	datetime	☑
rowguid	uniqueidentifier	☐
ModifiedDate	datetime	☐

- Finding a row by using a simple equality

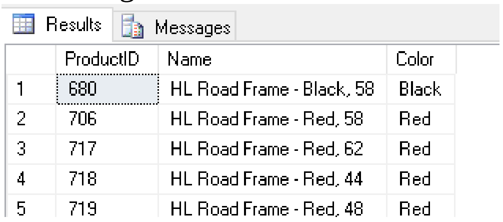
```
SELECT ProductID, Name
FROM Production.Product
WHERE Name = 'Blade' ;
```



	ProductID	Name
1	316	Blade

- Finding rows that contain a value as a part of a string

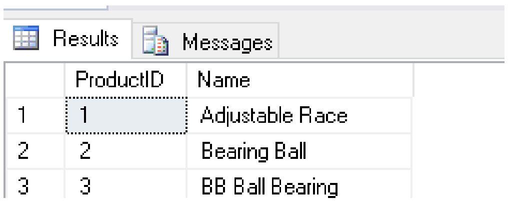
```
SELECT ProductID, Name, Color
FROM Production.Product
WHERE Name LIKE ('%Frame%');
```



	ProductID	Name	Color
1	680	HL Road Frame - Black, 58	Black
2	706	HL Road Frame - Red, 58	Red
3	717	HL Road Frame - Red, 62	Red
4	718	HL Road Frame - Red, 44	Red
5	719	HL Road Frame - Red, 48	Red

- Finding rows by using a comparison operator

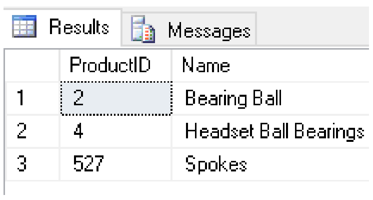
```
SELECT ProductID, Name
FROM Production.Product
WHERE ProductID <= 12 ;
```



	ProductID	Name
1	1	Adjustable Race
2	2	Bearing Ball
3	3	BB Ball Bearing

- Finding rows that meet any of three conditions

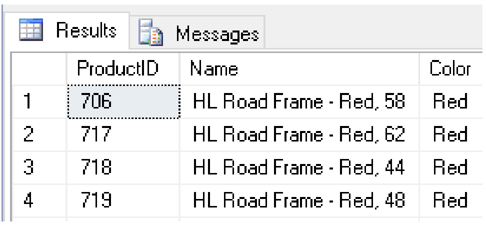
```
SELECT ProductID, Name
FROM Production.Product
WHERE ProductID = 2  OR ProductID = 4  OR Name = 'Spokes' ;
```



	ProductID	Name
1	2	Bearing Ball
2	4	Headset Ball Bearings
3	527	Spokes

- Finding rows that must meet several conditions

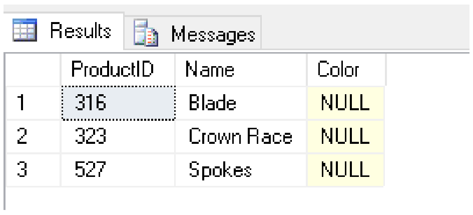
```
SELECT ProductID, Name, Color
FROM Production.Product
WHERE Name LIKE ('%Frame%')
AND Name LIKE ('HL%')  AND Color = 'Red' ;
```


Results | Messages

	ProductID	Name	Color
1	706	HL Road Frame - Red, 58	Red
2	717	HL Road Frame - Red, 62	Red
3	718	HL Road Frame - Red, 44	Red
4	719	HL Road Frame - Red, 48	Red

- Finding rows that are in a list of values

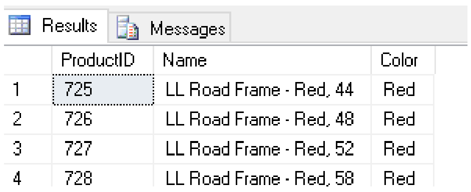
```
SELECT ProductID, Name, Color
FROM Production.Product
WHERE Name IN ('Blade', 'Crown Race', 'Spokes');
```


Results | Messages

	ProductID	Name	Color
1	316	Blade	NULL
2	323	Crown Race	NULL
3	527	Spokes	NULL

- Finding rows that have a value between two values

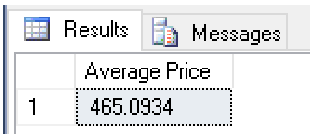
```
SELECT ProductID, Name, Color
FROM Production.Product
WHERE ProductID BETWEEN 725 AND 734;
```


Results | Messages

	ProductID	Name	Color
1	725	LL Road Frame - Red, 44	Red
2	726	LL Road Frame - Red, 48	Red
3	727	LL Road Frame - Red, 52	Red
4	728	LL Road Frame - Red, 58	Red

- With function

```
SELECT AVG(UnitPrice) AS [Average Price]
FROM Sales.SalesOrderDetail;
```

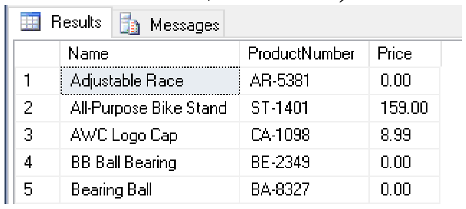

Results | Messages

	Average Price
1	465.0934

ORDER BY

- returns all rows and only a subset of the columns (Name, ProductNumber, ListPrice)

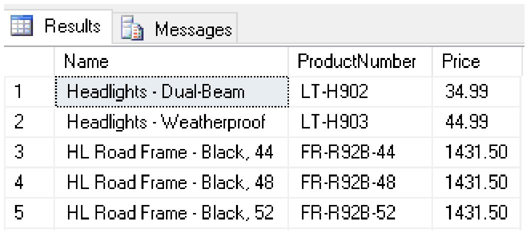
```
SELECT Name, ProductNumber, ListPrice AS Price
FROM Production.Product
ORDER BY Name ASC;
```


Results | Messages

	Name	ProductNumber	Price
1	Adjustable Race	AR-5381	0.00
2	All-Purpose Bike Stand	ST-1401	159.00
3	AWC Logo Cap	CA-1098	8.99
4	BB Ball Bearing	BE-2349	0.00
5	Bearing Ball	BA-8327	0.00

- returns only the rows for Product that have a product line of R and that have days to manufacture that is less than 4.

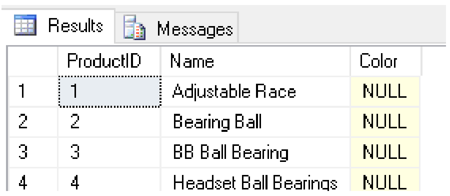
```
SELECT Name, ProductNumber, ListPrice AS Price
FROM Production.Product
WHERE ProductLine = 'R'  AND DaysToManufacture < 4
ORDER BY Name ASC;
```


Results | Messages

	Name	ProductNumber	Price
1	Headlights - Dual-Beam	LT-H902	34.99
2	Headlights - Weatherproof	LT-H903	44.99
3	HL Road Frame - Black, 44	FR-R92B-44	1431.50
4	HL Road Frame - Black, 48	FR-R92B-48	1431.50
5	HL Road Frame - Black, 52	FR-R92B-52	1431.50

- orders the result set by a column that is not included in the select list, but is defined in the table specified in the FROM clause.

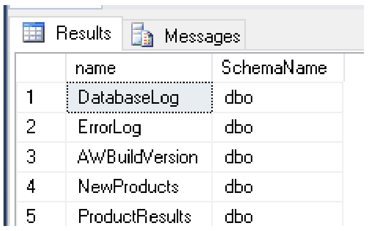
```
SELECT ProductID, Name, Color
FROM Production.Product
ORDER BY ListPrice;
```


Results | Messages

	ProductID	Name	Color
1	1	Adjustable Race	NULL
2	2	Bearing Ball	NULL
3	3	BB Ball Bearing	NULL
4	4	Headset Ball Bearings	NULL

- specifies the column alias SchemaName as the sort order column.

```
SELECT name, SCHEMA_NAME(schema_id) AS
SchemaName
FROM sys.objects
WHERE type = 'U'
ORDER BY SchemaName;
```

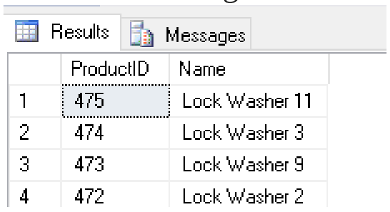


Results | Messages

	name	SchemaName
1	DatabaseLog	dbo
2	ErrorLog	dbo
3	AWBuildVersion	dbo
4	NewProducts	dbo
5	ProductResults	dbo

- orders the result set by the numeric column ProductID in descending order.

```
SELECT ProductID, Name
FROM Production.Product
WHERE Name LIKE 'Lock Washer%'
ORDER BY ProductID DESC;
```

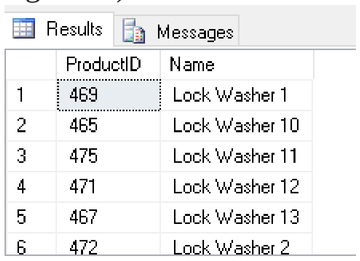


Results | Messages

	ProductID	Name
1	475	Lock Washer 11
2	474	Lock Washer 3
3	473	Lock Washer 9
4	472	Lock Washer 2

- orders the result set by the Name column in ascending order(the characters are sorted alphabetically, 10 sorts before 2) - (default - ascending order)

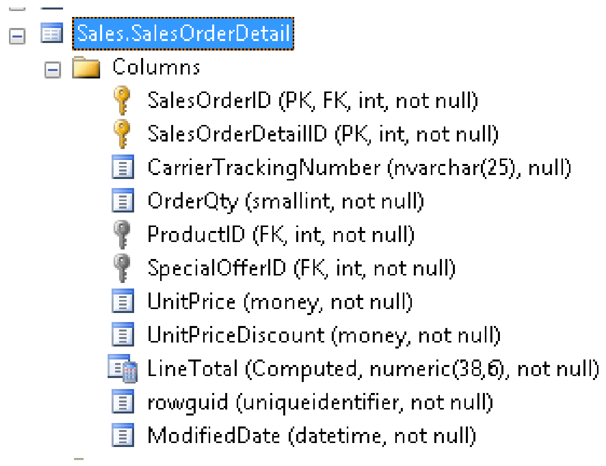
```
SELECT ProductID, Name
FROM Production.Product
WHERE Name LIKE 'Lock Washer%'
ORDER BY Name ASC ;
```



Results | Messages

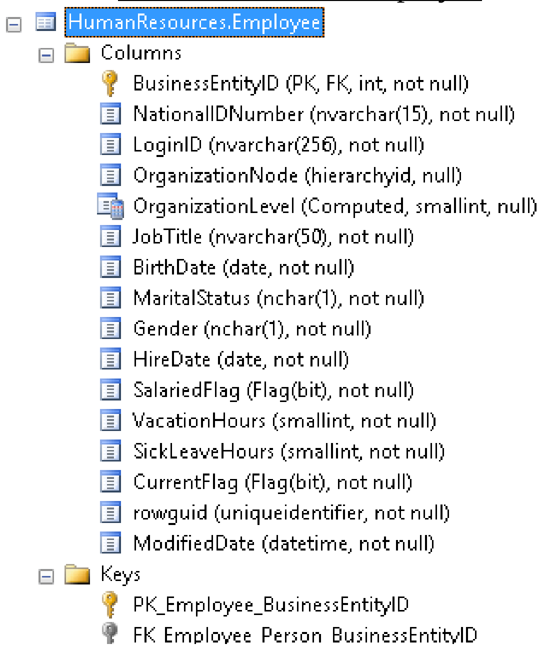
	ProductID	Name
1	469	Lock Washer 1
2	465	Lock Washer 10
3	475	Lock Washer 11
4	471	Lock Washer 12
5	467	Lock Washer 13
6	472	Lock Washer 2

Table Sales.SalesOrderDetail



```
Sales.SalesOrderDetail
  Columns
    SalesOrderID (PK, FK, int, not null)
    SalesOrderDetailID (PK, int, not null)
    CarrierTrackingNumber (nvarchar(25), null)
    OrderQty (smallint, not null)
    ProductID (FK, int, not null)
    SpecialOfferID (FK, int, not null)
    UnitPrice (money, not null)
    UnitPriceDiscount (money, not null)
    LineTotal (Computed, numeric(38,6), not null)
    rowguid (uniqueidentifier, not null)
    ModifiedDate (datetime, not null)
```

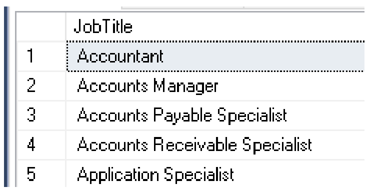
Table HumanResources.Employee



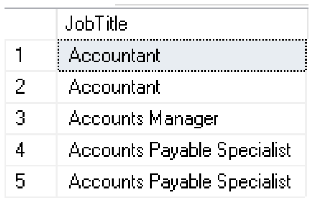
```
HumanResources.Employee
  Columns
    BusinessEntityID (PK, FK, int, not null)
    NationalIDNumber (nvarchar(15), not null)
    LoginID (nvarchar(256), not null)
    OrganizationNode (hierarchyid, null)
    OrganizationLevel (Computed, smallint, null)
    JobTitle (nvarchar(50), not null)
    BirthDate (date, not null)
    MaritalStatus (nchar(1), not null)
    Gender (nchar(1), not null)
    HireDate (date, not null)
    SalariedFlag (Flag(bit), not null)
    VacationHours (smallint, not null)
    SickLeaveHours (smallint, not null)
    CurrentFlag (Flag(bit), not null)
    rowguid (uniqueidentifier, not null)
    ModifiedDate (datetime, not null)
  Keys
    PK_Employee_BusinessEntityID
    FK_Employee_Person_BusinessEntityID
```

- uses DISTINCT to prevent the retrieval of duplicate titles.
- uses TOP to select only the first rows from the result set

```
SELECT DISTINCT JobTitle
FROM HumanResources.Employee
ORDER BY JobTitle;
   SELECT TOP 5 JobTitle
   FROM HumanResources.Employee
   ORDER BY JobTitle;
```



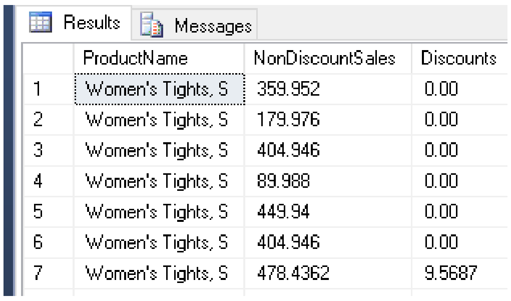
	JobTitle
1	Accountant
2	Accounts Manager
3	Accounts Payable Specialist
4	Accounts Receivable Specialist
5	Application Specialist



	JobTitle
1	Accountant
2	Accountant
3	Accounts Manager
4	Accounts Payable Specialist
5	Accounts Payable Specialist

- returns total sales and the discounts for each product.

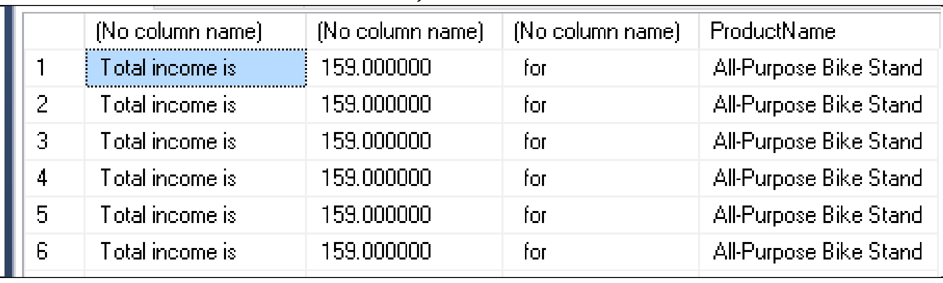
```
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName DESC;
```



	ProductName	NonDiscountSales	Discounts
1	Women's Tights, S	359.952	0.00
2	Women's Tights, S	179.976	0.00
3	Women's Tights, S	404.946	0.00
4	Women's Tights, S	89.988	0.00
5	Women's Tights, S	449.94	0.00
6	Women's Tights, S	404.946	0.00
7	Women's Tights, S	478.4362	9.5687

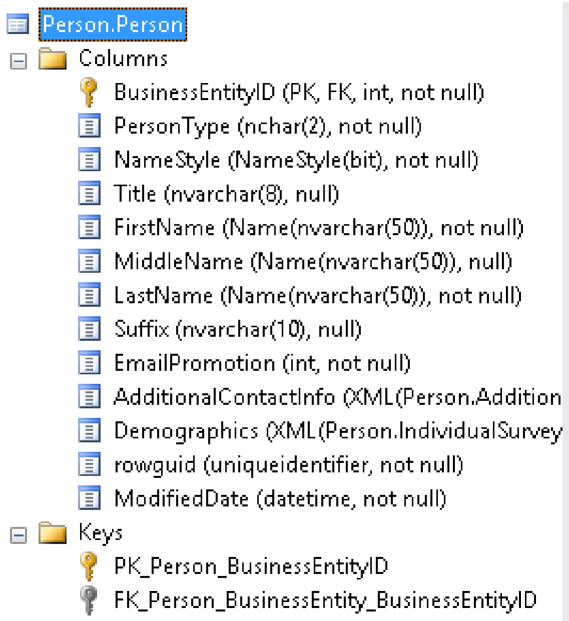
- calculates the revenue for each product in each sales order.

```
SELECT 'Total income is', ((OrderQty * UnitPrice) * (1.0 - UnitPriceDiscount)), ' for ',
p.Name AS ProductName
FROM Production.Product AS p INNER JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName ASC;
```



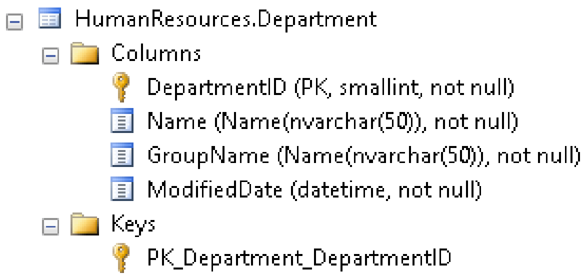
	(No column name)	(No column name)	(No column name)	ProductName
1	Total income is	159.000000	for	All-Purpose Bike Stand
2	Total income is	159.000000	for	All-Purpose Bike Stand
3	Total income is	159.000000	for	All-Purpose Bike Stand
4	Total income is	159.000000	for	All-Purpose Bike Stand
5	Total income is	159.000000	for	All-Purpose Bike Stand
6	Total income is	159.000000	for	All-Purpose Bike Stand

Table Person.Person



```
Person.Person
  Columns
    BusinessEntityID (PK, FK, int, not null)
    PersonType (nchar(2), not null)
    NameStyle (NameStyle(bit), not null)
    Title (nvarchar(8), null)
    FirstName (Name(nvarchar(50)), not null)
    MiddleName (Name(nvarchar(50)), null)
    LastName (Name(nvarchar(50)), not null)
    Suffix (nvarchar(10), null)
    EmailPromotion (int, not null)
    AdditionalContactInfo (XML(Person.Addition
    Demographics (XML(Person.IndividualSurvey
    rowguid (uniqueidentifier, not null)
    ModifiedDate (datetime, not null)
  Keys
    PK_Person_BusinessEntityID
    FK_Person_BusinessEntity_BusinessEntityID
```

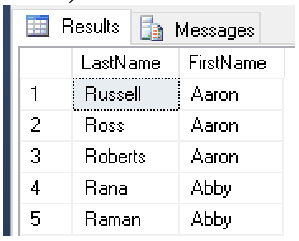
Table HumanResources.Department



```
HumanResources.Department
  Columns
    DepartmentID (PK, smallint, not null)
    Name (Name(nvarchar(50)), not null)
    GroupName (Name(nvarchar(50)), not null)
    ModifiedDate (datetime, not null)
  Keys
    PK_Department_DepartmentID
```

- orders the result set by two columns(first sorted in ascending order by the FirstName column and then sorted in descending order by the LastName column)

```
SELECT LastName, FirstName
FROM Person.Person
WHERE LastName LIKE 'R%'
ORDER BY FirstName ASC, LastName DESC ;
```



	LastName	FirstName
1	Russell	Aaron
2	Ross	Aaron
3	Roberts	Aaron
4	Rana	Abby
5	Raman	Abby

-use OFFSET and FETCH to limit the number of rows returned by a query.

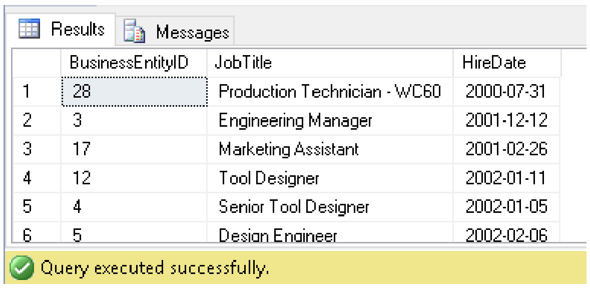
```
-- Return all rows sorted by the column DepartmentID.
SELECT DepartmentID, Name, GroupName
FROM HumanResources.Department
ORDER BY DepartmentID;

-- Skip the first 5 rows from the sorted result set and return all remaining rows.
SELECT DepartmentID, Name, GroupName
FROM HumanResources.Department
ORDER BY DepartmentID OFFSET 5 ROWS;

-- Skip 0 rows and return only the first 10 rows from the sorted result set.
SELECT DepartmentID, Name, GroupName
FROM HumanResources.Department
ORDER BY DepartmentID   OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;
```

- uses an expression as the sort column, defined by using the DATEPART function to sort the result set by the year in which employees were hired.

```
SELECT BusinessEntityID, JobTitle, HireDate
FROM HumanResources.Employee
ORDER BY DATEPART(year, HireDate);
```



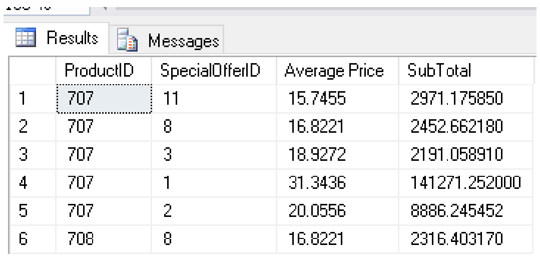
	BusinessEntityID	JobTitle	HireDate
1	28	Production Technician - WC60	2000-07-31
2	3	Engineering Manager	2001-12-12
3	17	Marketing Assistant	2001-02-26
4	12	Tool Designer	2002-01-11
5	4	Senior Tool Designer	2002-01-05
6	5	Design Engineer	2002-02-06

Query executed successfully.

GROUP BY

- finds the average price and the sum of year-to-date sales, grouped by product ID and special offer ID.

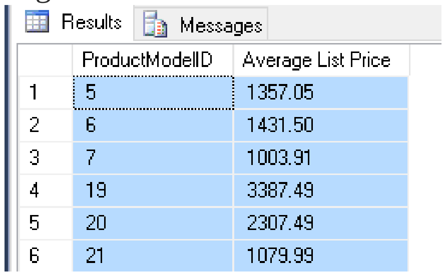
```
SELECT ProductID, SpecialOfferID, AVG(UnitPrice)
AS [Average Price],  SUM(LineTotal) AS SubTotal
FROM Sales.SalesOrderDetail
GROUP BY ProductID, SpecialOfferID
ORDER BY ProductID;
```



	ProductID	SpecialOfferID	Average Price	SubTotal
1	707	11	15.7455	2971.175850
2	707	8	16.8221	2452.662180
3	707	3	18.9272	2191.058910
4	707	1	31.3436	141271.252000
5	707	2	20.0556	8886.245452
6	708	8	16.8221	2316.403170

- puts the results into groups after retrieving only the rows with list prices greater than $1000.

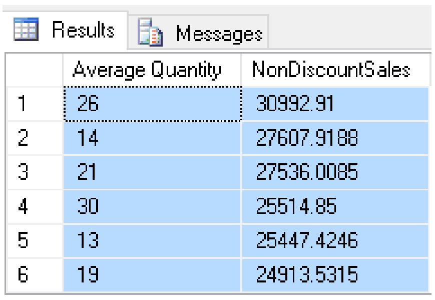
```
SELECT ProductModelID, AVG(ListPrice) AS [Average List Price]
FROM Production.Product
WHERE ListPrice > $1000
GROUP BY ProductModelID
ORDER BY ProductModelID;
```



	ProductModelID	Average List Price
1	5	1357.05
2	6	1431.50
3	7	1003.91
4	19	3387.49
5	20	2307.49
6	21	1079.99

- groups by an expression (if the expression does not include aggregate functions).

```
SELECT AVG(OrderQty) AS [Average Quantity],
NonDiscountSales = (OrderQty * UnitPrice)
FROM Sales.SalesOrderDetail
GROUP BY (OrderQty * UnitPrice)
ORDER BY (OrderQty * UnitPrice) DESC;
```

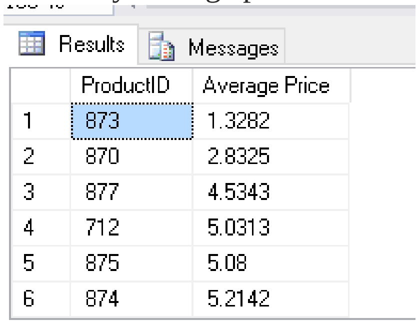


Results | Messages

	Average Quantity	NonDiscountSales
1	26	30992.91
2	14	27607.9188
3	21	27536.0085
4	30	25514.85
5	13	25447.4246
6	19	24913.5315

- finds the average price of each type of product and orders the results by average price.

```
SELECT ProductID, AVG(UnitPrice) AS [Average Price]
FROM Sales.SalesOrderDetail
WHERE OrderQty > 10
GROUP BY ProductID
ORDER BY AVG(UnitPrice);
```



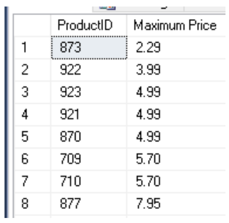
Results | Messages

	ProductID	Average Price
1	873	1.3282
2	870	2.8325
3	877	4.5343
4	712	5.0313
5	875	5.08
6	874	5.2142

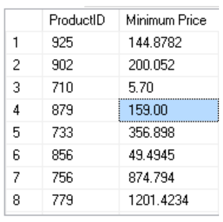
- finds the maximum/minimim price of products

```
SELECT ProductID, MAX(UnitPrice) AS [Maximum Price]
FROM Sales.SalesOrderDetail
GROUP BY ProductID
ORDER BY MAX(UnitPrice);

SELECT ProductID, MIN(UnitPrice) AS [Minimum Price]
FROM Sales.SalesOrderDetail
GROUP BY ProductID
```



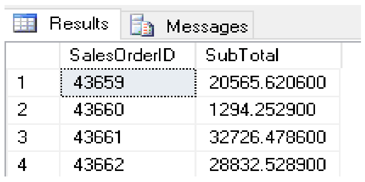
	ProductID	Maximum Price
1	873	2.29
2	922	3.99
3	923	4.99
4	921	4.99
5	870	4.99
6	709	5.70
7	710	5.70
8	877	7.95



	ProductID	Minimum Price
1	925	144.8782
2	902	200.052
3	710	5.70
4	879	159.00
5	733	356.898
6	856	49.4945
7	756	874.794
8	779	1201.4234

- retrieves the total for each SalesOrderID from the SalesOrderDetail table.

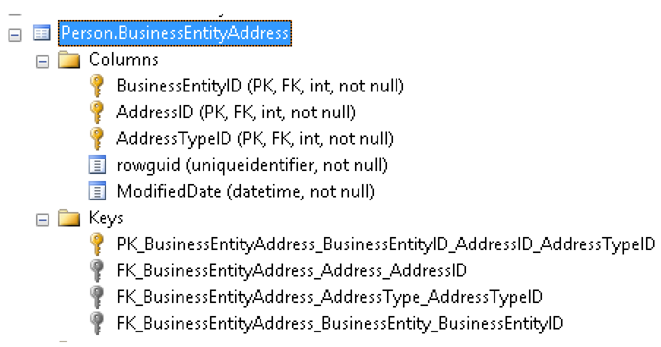
```
SELECT SalesOrderID, SUM(LineTotal) AS SubTotal
FROM Sales.SalesOrderDetail AS sod
GROUP BY SalesOrderID
ORDER BY SalesOrderID;
```



Results | Messages

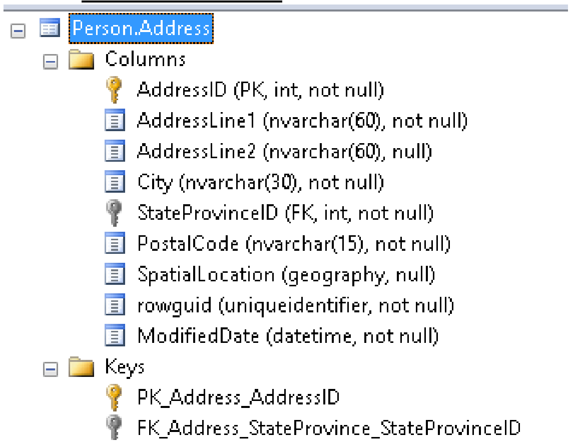
	SalesOrderID	SubTotal
1	43659	20565.620600
2	43660	1294.252900
3	43661	32726.478600
4	43662	28832.528900

Table Person.Business.EntityAddress



```
Person.BusinessEntityAddress
  Columns
    BusinessEntityID (PK, FK, int, not null)
    AddressID (PK, FK, int, not null)
    AddressTypeID (PK, FK, int, not null)
    rowguid (uniqueidentifier, not null)
    ModifiedDate (datetime, not null)
  Keys
    PK_BusinessEntityAddress_BusinessEntityID_AddressID_AddressTypeID
    FK_BusinessEntityAddress_Address_AddressID
    FK_BusinessEntityAddress_AddressType_AddressTypeID
    FK_BusinessEntityAddress_BusinessEntity_BusinessEntityID
```

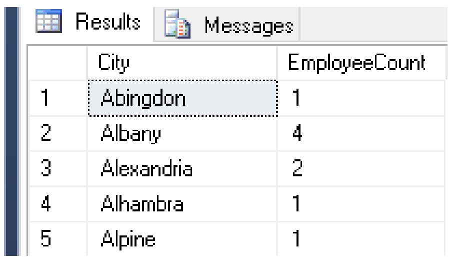
Table Person.Address



```
Person.Address
  Columns
    AddressID (PK, int, not null)
    AddressLine1 (nvarchar(60), not null)
    AddressLine2 (nvarchar(60), null)
    City (nvarchar(30), not null)
    StateProvinceID (FK, int, not null)
    PostalCode (nvarchar(15), not null)
    SpatialLocation (geography, null)
    rowguid (uniqueidentifier, not null)
    ModifiedDate (datetime, not null)
  Keys
    PK_Address_AddressID
    FK_Address_StateProvince_StateProvinceID
```

- retrieves the number of employees for each City from the Address table joined to the EmployeeAddress table.

```
SELECT a.City, COUNT(bea.AddressID) EmployeeCount
FROM Person.BusinessEntityAddress AS bea
    INNER JOIN Person.Address AS a
        ON bea.AddressID = a.AddressID
GROUP BY a.City
ORDER BY a.City;
```

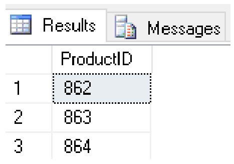


	City	EmployeeCount
1	Abingdon	1
2	Albany	4
3	Alexandria	2
4	Alhambra	1
5	Alpine	1

HAVING

- a HAVING clause with an aggregate function. It groups the rows in the SalesOrderDetail table by product ID and eliminates products whose average order quantities are five or less.

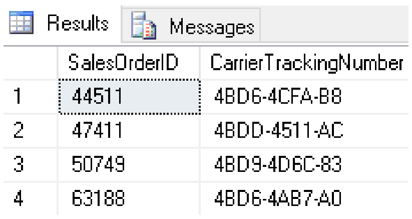
```
SELECT ProductID
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING AVG(OrderQty) > 5
ORDER BY ProductID;
```



	ProductID
1	862
2	863
3	864

- a HAVING clause without aggregate functions, with the LIKE clause in the HAVING clause.

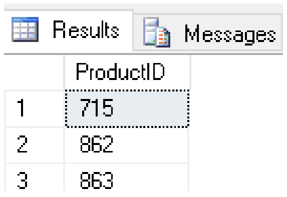
```
SELECT SalesOrderID, CarrierTrackingNumber
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID,
CarrierTrackingNumber
HAVING CarrierTrackingNumber LIKE '4BD%'
ORDER BY SalesOrderID ;
```



	SalesOrderID	CarrierTrackingNumber
1	44511	4BD6-4CFA-B8
2	47411	4BDD-4511-AC
3	50749	4BD9-4D6C-83
4	63188	4BD6-4AB7-A0

- groups and summary values, after eliminating the products with prices over $25 and average order quantities under 5, organized byProductID.

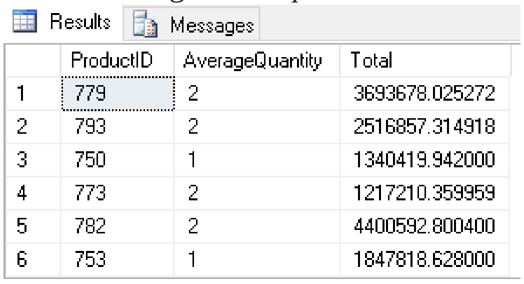
```
SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice < 25.00
GROUP BY ProductID
HAVING AVG(OrderQty) > 5
ORDER BY ProductID;
```



	ProductID
1	715
2	862
3	863

- groups the SalesOrderDetail table by product ID and includes only those groups of products that have orders totaling more than $1000000.00 and whose average order quantities are less than 3.

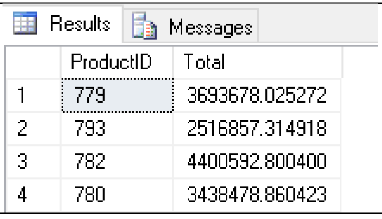
```
SELECT ProductID, AVG(OrderQty) AS
AverageQuantity, SUM(LineTotal) AS Total
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING SUM(LineTotal) > $1000000.00
AND AVG(OrderQty) < 3;
```



	ProductID	AverageQuantity	Total
1	779	2	3693678.025272
2	793	2	2516857.314918
3	750	1	1340419.942000
4	773	2	1217210.359959
5	782	2	4400592.800400
6	753	1	1847818.628000

- the products that have had total sales greater than $2000000.00

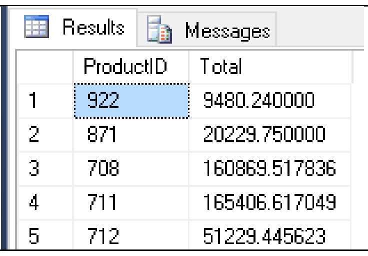
```
SELECT ProductID, Total = SUM(LineTotal)
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING SUM(LineTotal) > $2000000.00;
```



	ProductID	Total
1	779	3693678.025272
2	793	2516857.314918
3	782	4400592.800400
4	780	3438478.860423

- to make sure there are at least 1500 items involved in the calculations for each product, use HAVING COUNT(*) > 1500 to eliminate the products that return totals for fewer than 1500 items sold.

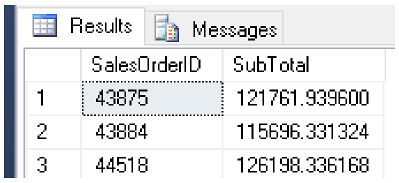
```
SELECT ProductID, SUM(LineTotal) AS Total
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING COUNT(*) > 1500;
```



	ProductID	Total
1	922	9480.240000
2	871	20229.750000
3	708	160869.517836
4	711	165406.617049
5	712	51229.445623

- uses a HAVING clause retrieves the total for each SalesOrderID from the SalesOrderDetail table that exceeds $100000.00.

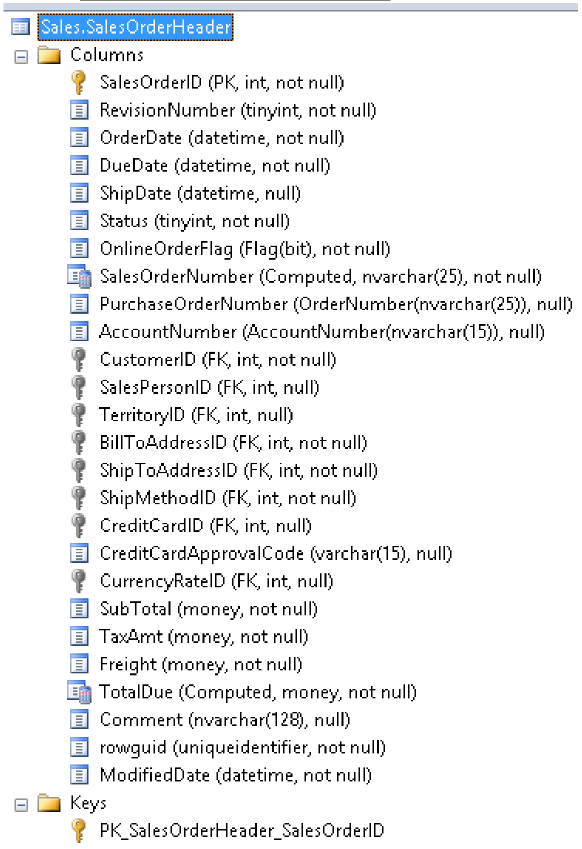
```
SELECT SalesOrderID, SUM(LineTotal) AS SubTotal
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
HAVING SUM(LineTotal) > 100000.00
ORDER BY SalesOrderID ;
```



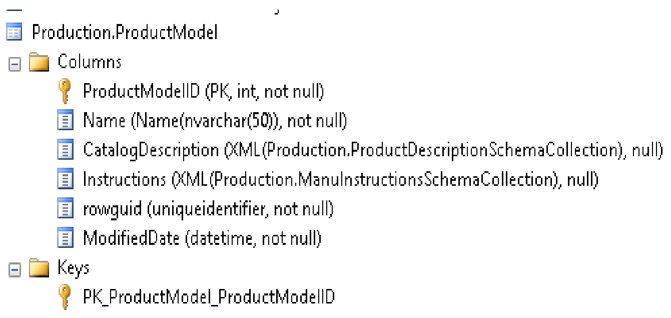
	SalesOrderID	SubTotal
1	43875	121761.939600
2	43884	115696.331324
3	44518	126198.336168

Table Sales.SalesOrderHeader

Table Production.ProductModel



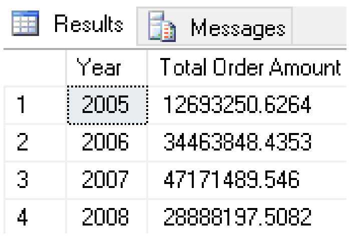
Sales.SalesOrderHeader
Columns
SalesOrderID (PK, int, not null)
RevisionNumber (tinyint, not null)
OrderDate (datetime, not null)
DueDate (datetime, not null)
ShipDate (datetime, null)
Status (tinyint, not null)
OnlineOrderFlag (Flag(bit), not null)
SalesOrderNumber (Computed, nvarchar(25), not null)
PurchaseOrderNumber (OrderNumber(nvarchar(25)), null)
AccountNumber (AccountNumber(nvarchar(15)), null)
CustomerID (FK, int, not null)
SalesPersonID (FK, int, null)
TerritoryID (FK, int, null)
BillToAddressID (FK, int, not null)
ShipToAddressID (FK, int, not null)
ShipMethodID (FK, int, not null)
CreditCardID (FK, int, null)
CreditCardApprovalCode (varchar(15), null)
CurrencyRateID (FK, int, null)
SubTotal (money, not null)
TaxAmt (money, not null)
Freight (money, not null)
TotalDue (Computed, money, not null)
Comment (nvarchar(128), null)
rowguid (uniqueidentifier, not null)
ModifiedDate (datetime, not null)
Keys
PK_SalesOrderHeader_SalesOrderID



Production.ProductModel
Columns
ProductModelID (PK, int, not null)
Name (Name(nvarchar(50)), not null)
CatalogDescription (XML(Production.ProductDescriptionSchemaCollection), null)
Instructions (XML(Production.ManuInstructionsSchemaCollection), null)
rowguid (uniqueidentifier, not null)
ModifiedDate (datetime, not null)
Keys
PK_ProductModel_ProductModelID

- uses the HAVING clause to specify which of the groups generated in the GROUP BY clause should be included in the result set.

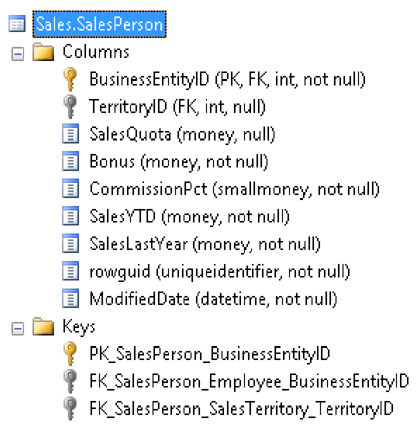
```
SELECT DATEPART(yyyy,OrderDate) AS N'Year',
  SUM(TotalDue) AS N'Total Order Amount'
FROM Sales.SalesOrderHeader
GROUP BY DATEPART(yyyy,OrderDate)
HAVING DATEPART(yyyy,OrderDate) >= N'2003'
ORDER BY DATEPART(yyyy,OrderDate);
```



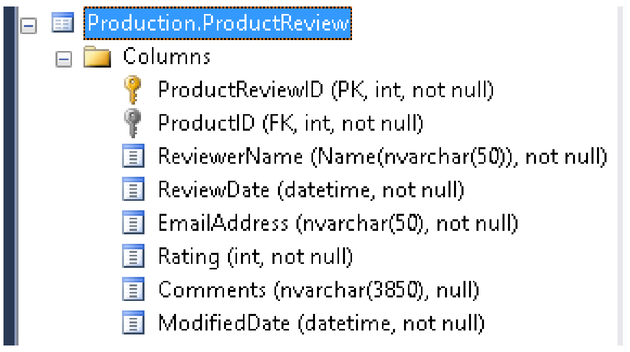
	Year	Total Order Amount
1	2005	12693250.6264
2	2006	34463848.4353
3	2007	47171489.546
4	2008	28888197.5082

Table Sales.SalesPerson

Table Production.ProductReview



```
Sales.SalesPerson
  Columns
    BusinessEntityID (PK, FK, int, not null)
    TerritoryID (FK, int, null)
    SalesQuota (money, null)
    Bonus (money, not null)
    CommissionPct (smallmoney, not null)
    SalesYTD (money, not null)
    SalesLastYear (money, not null)
    rowguid (uniqueidentifier, not null)
    ModifiedDate (datetime, not null)
  Keys
    PK_SalesPerson_BusinessEntityID
    FK_SalesPerson_Employee_BusinessEntityID
    FK_SalesPerson_SalesTerritory_TerritoryID
```

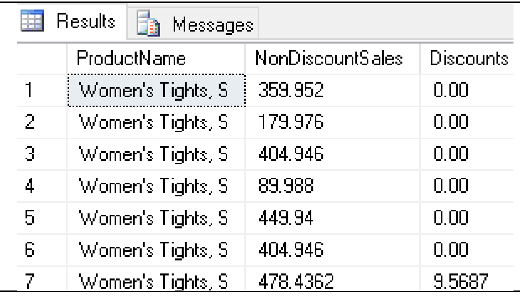


```
Production.ProductReview
  Columns
    ProductReviewID (PK, int, not null)
    ProductID (FK, int, not null)
    ReviewerName (Name(nvarchar(50)), not null)
    ReviewDate (datetime, not null)
    EmailAddress (nvarchar(50), not null)
    Rating (int, not null)
    Comments (nvarchar(3850), null)
    ModifiedDate (datetime, not null)
```

INNER JOIN

- Return all rows and the columns that calculate the total sales and the discount for each product from the Product table ordered by the names of the products.

```
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) *
UnitPriceDiscount) FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName DESC;
```

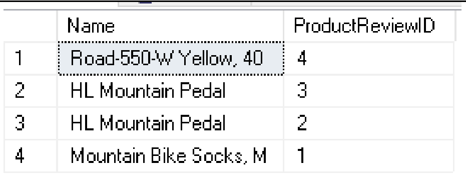


Results | Messages

	ProductName	NonDiscountSales	Discounts
1	Women's Tights, S	359.952	0.00
2	Women's Tights, S	179.976	0.00
3	Women's Tights, S	404.946	0.00
4	Women's Tights, S	89.988	0.00
5	Women's Tights, S	449.94	0.00
6	Women's Tights, S	404.946	0.00
7	Women's Tights, S	478.4362	9.5687

- inner join

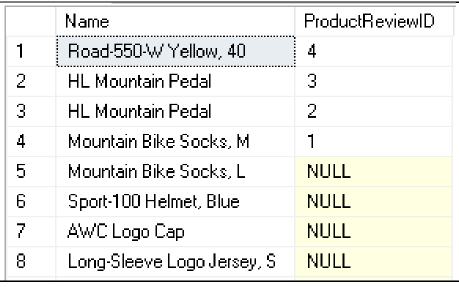
```
SELECT p.Name, pr.ProductReviewID
FROM Production.Product AS p
INNER JOIN Production.ProductReview AS pr
ON p.ProductID = pr.ProductID
ORDER BY ProductReviewID DESC;
```



	Name	ProductReviewID
1	Road-550-W Yellow, 40	4
2	HL Mountain Pedal	3
3	HL Mountain Pedal	2
4	Mountain Bike Socks, M	1

- left outer join

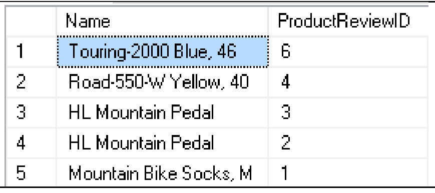
```
SELECT p.Name, pr.ProductReviewID
FROM Production.Product AS p
LEFT OUTER JOIN Production.ProductReview AS pr
ON p.ProductID = pr.ProductID
ORDER BY ProductReviewID DESC;
```



	Name	ProductReviewID
1	Road-550-W Yellow, 40	4
2	HL Mountain Pedal	3
3	HL Mountain Pedal	2
4	Mountain Bike Socks, M	1
5	Mountain Bike Socks, L	NULL
6	Sport-100 Helmet, Blue	NULL
7	AWC Logo Cap	NULL
8	Long-Sleeve Logo Jersey, S	NULL

- right outer join

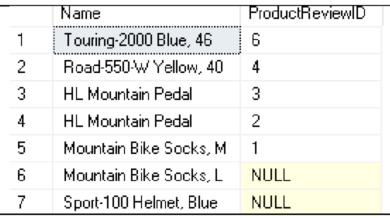
```
SELECT p.Name, pr.ProductReviewID
FROM Production.Product AS p
RIGHT OUTER JOIN Production.ProductReview AS pr
ON p.ProductID = pr.ProductID
ORDER BY ProductReviewID DESC;
```



	Name	ProductReviewID
1	Touring-2000 Blue, 46	6
2	Road-550-W Yellow, 40	4
3	HL Mountain Pedal	3
4	HL Mountain Pedal	2
5	Mountain Bike Socks, M	1

- full outer join

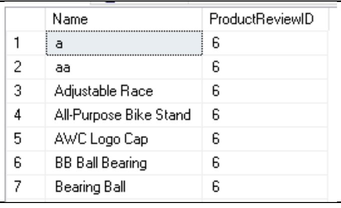
```
SELECT p.Name, pr.ProductReviewID
FROM Production.Product AS p
FULL OUTER JOIN Production.ProductReview AS pr
ON p.ProductID = pr.ProductID
ORDER BY ProductReviewID DESC;
```



	Name	ProductReviewID
1	Touring-2000 Blue, 46	6
2	Road-550-W Yellow, 40	4
3	HL Mountain Pedal	3
4	HL Mountain Pedal	2
5	Mountain Bike Socks, M	1
6	Mountain Bike Socks, L	NULL
7	Sport-100 Helmet, Blue	NULL

- cross join

```
SELECT p.Name, pr.ProductReviewID
FROM Production.Product AS p
CROSS JOIN Production.ProductReview AS pr
ORDER BY ProductReviewID DESC;
```

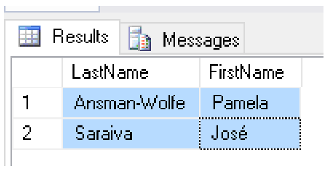


	Name	ProductReviewID
1	a	6
2	aa	6
3	Adjustable Race	6
4	All-Purpose Bike Stand	6
5	AWC Logo Cap	6
6	BB Ball Bearing	6
7	Bearing Ball	6

<u>QUERY IN QUERY</u>

- retrieves one instance of the first and last name of each employee for which the bonus in the SalesPerson table is 5000.00 and for which the employee identification numbers match in the Employee and SalesPerson tables. The subquery cannot be evaluated independently of the outer query (it requires a value for Employee.EmployeeID, but this value changes as the SQL Server Database Engine examines different rows in Employee).

```
SELECT DISTINCT p.LastName, p.FirstName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
   ON e.BusinessEntityID = p.BusinessEntityID WHERE 5000.00 IN
   (SELECT Bonus
    FROM Sales.SalesPerson AS sp
    WHERE e.BusinessEntityID = sp.BusinessEntityID);
```



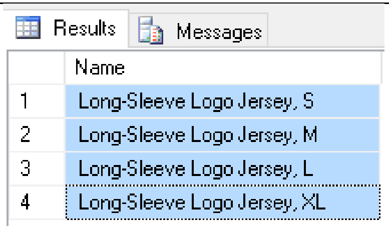
Results | Messages

	LastName	FirstName
1	Ansman-Wolfe	Pamela
2	Saraiva	José

- Retrieve one instance of each product name for which the product model is a long sleeve logo jersey, and the ProductModelID numbers match between the Product and ProductModel tables.

```
SELECT DISTINCT Name
FROM Production.Product AS p
WHERE EXISTS
(SELECT *
FROM Production.ProductModel AS pm
WHERE p.ProductModelID = pm.ProductModelID
AND pm.Name LIKE 'Long-Sleeve Logo Jersey%');
```

```
SELECT DISTINCT Name
FROM Production.Product
WHERE ProductModelID IN
(SELECT ProductModelID
FROM Production.ProductModel
WHERE Name LIKE 'Long-Sleeve Logo Jersey%');
```



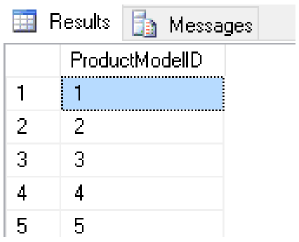
Results | Messages

	Name
1	Long-Sleeve Logo Jersey, S
2	Long-Sleeve Logo Jersey, M
3	Long-Sleeve Logo Jersey, L
4	Long-Sleeve Logo Jersey, XL

- a subquery can be used in the HAVING clause of an outer query - finds the product models for which the maximum list price is more than twice the average for the model.

```
SELECT p1.ProductModelID
FROM Production.Product AS p1
GROUP BY p1.ProductModelID
HAVING MAX(p1.ListPrice) >= ALL
   (SELECT AVG(p2.ListPrice)
    FROM Production.Product AS p2
    WHERE p1.ProductModelID = p2.ProductModelID);
```

```
SELECT p1.ProductModelID
FROM Production.Product AS p1
GROUP BY p1.ProductModelID
HAVING MAX(p1.ListPrice) >= ANY
(SELECT AVG(p2.ListPrice)
FROM Production.Product AS p2
```



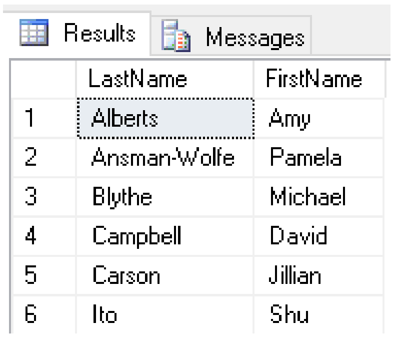
Results | Messages

	ProductModelID
1	1
2	2
3	3
4	4
5	5

```
WHERE p1.ProductModelID = p2.ProductModelID);
```

- uses two correlated subqueries to find the names of employees who have sold a particular product.

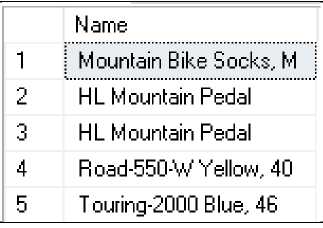
```
SELECT DISTINCT pp.LastName, pp.FirstName
FROM Person.Person pp JOIN HumanResources.Employee e
ON e.BusinessEntityID = pp.BusinessEntityID
WHERE pp.BusinessEntityID IN
(SELECT SalesPersonID
FROM Sales.SalesOrderHeader
WHERE SalesOrderID IN
(SELECT SalesOrderID
FROM Sales.SalesOrderDetail
WHERE ProductID IN
(SELECT ProductID
FROM Production.Product p
WHERE ProductNumber = 'BK-M68B-42')));
```



	LastName	FirstName
1	Alberts	Amy
2	Ansman-Wolfe	Pamela
3	Blythe	Michael
4	Campbell	David
5	Carson	Jillian
6	Ito	Shu

- uses in from

```
SELECT a.Name
FROM (SELECT p.Name, pr.ProductReviewID
FROM Production.Product AS p
INNER JOIN Production.ProductReview AS pr
ON p.ProductID = pr.ProductID) a
```

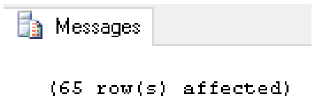


	Name
1	Mountain Bike Socks, M
2	HL Mountain Pedal
3	HL Mountain Pedal
4	Road-550-W Yellow, 40
5	Touring-2000 Blue, 46

SELECT INTO

- creates the table NewProducts.

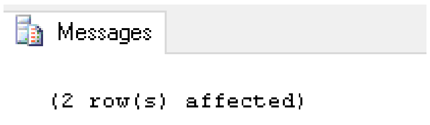
```
SELECT * INTO dbo.NewProducts
FROM Production.Product
WHERE ListPrice > $25 AND ListPrice < $100;
```



```
(65 row(s) affected)
```

- the result set includes the contents of the ProductModelID and Name columns of both the ProductModel and Glovestables.

```
IF OBJECT_ID ('dbo.Gloves', 'U') IS NOT NULL
DROP TABLE dbo.Gloves;
GO
-- Create Gloves table.
SELECT ProductModelID, Name
INTO dbo.Gloves
FROM Production.ProductModel
WHERE ProductModelID IN (3, 4);
GO
```

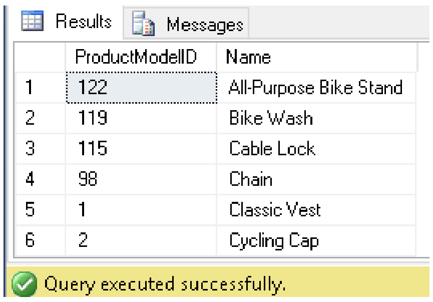


```
(2 row(s) affected)
```

UNION

- The simple union/intersect/except.

```
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID NOT IN (3, 4)
UNION
SELECT ProductModelID, Name
FROM dbo.Gloves
ORDER BY Name;
GO
```

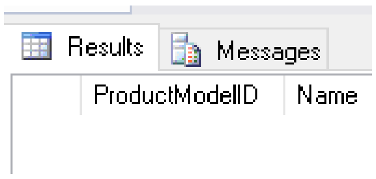


	ProductModelID	Name
1	122	All-Purpose Bike Stand
2	119	Bike Wash
3	115	Cable Lock
4	98	Chain
5	1	Classic Vest
6	2	Cycling Cap

Query executed successfully.

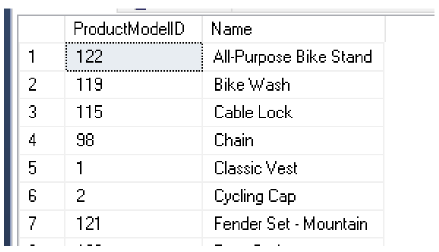
128 rows

```
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID NOT IN (3, 4)
INTERSECT
SELECT ProductModelID, Name
FROM dbo.Gloves
ORDER BY Name;
GO
```



	ProductModelID	Name

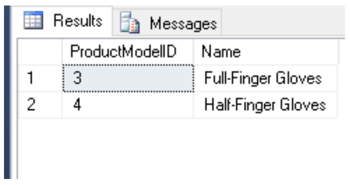
```
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID NOT IN (3, 4)
EXCEPT
SELECT ProductModelID, Name
FROM dbo.Gloves
ORDER BY Name;
GO
```



	ProductModelID	Name
1	122	All-Purpose Bike Stand
2	119	Bike Wash
3	115	Cable Lock
4	98	Chain
5	1	Classic Vest
6	2	Cycling Cap
7	121	Fender Set - Mountain

126 rows

```
SELECT ProductModelID, Name
FROM dbo.Gloves
EXCEPT
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID NOT IN (3, 4)
ORDER BY Name
GO
```



	ProductModelID	Name
1	3	Full-Finger Gloves
2	4	Half-Finger Gloves

2 rows

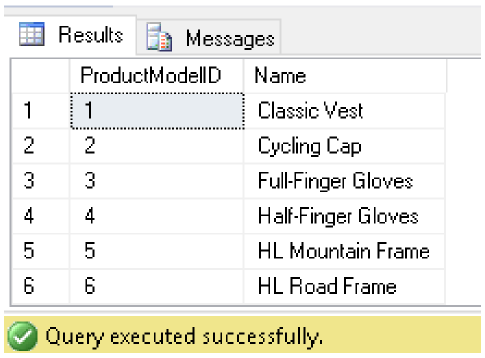
- the INTO clause in the second SELECT statement specifies that the table named ProductResults holds the final result set of the union of the designated columns of the ProductModel and Gloves tables. The Gloves table is created in the first SELECT statement.

```
IF OBJECT_ID ('dbo.ProductResults', 'U') IS NOT NULL
DROP TABLE dbo.ProductResults;
GO
IF OBJECT_ID ('dbo.Gloves', 'U') IS NOT NULL
DROP TABLE dbo.Gloves;
GO

SELECT ProductModelID, Name
INTO dbo.Gloves
FROM Production.ProductModel
WHERE ProductModelID IN (3, 4);
GO

SELECT ProductModelID, Name
INTO dbo.ProductResults
FROM Production.ProductModel
WHERE ProductModelID NOT IN (3, 4)
UNION
SELECT ProductModelID, Name
FROM dbo.Gloves;
GO

SELECT ProductModelID, Name
```



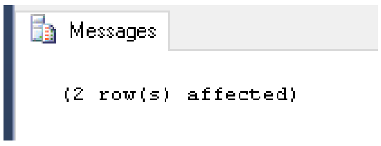
	ProductModelID	Name
1	1	Classic Vest
2	2	Cycling Cap
3	3	Full-Finger Gloves
4	4	Half-Finger Gloves
5	5	HL Mountain Frame
6	6	HL Road Frame

Query executed successfully.

```
FROM dbo.ProductResults;
```

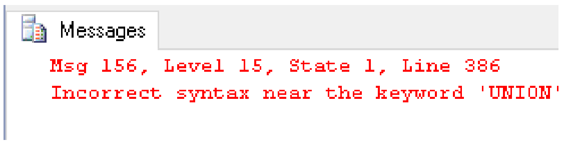
- The order of certain parameters used with the UNION clause is important.

```
IF OBJECT_ID ('dbo.Gloves', 'U') IS NOT NULL
DROP TABLE dbo.Gloves;
GO
SELECT ProductModelID, Name
INTO dbo.Gloves
FROM Production.ProductModel
WHERE ProductModelID IN (3, 4);
```



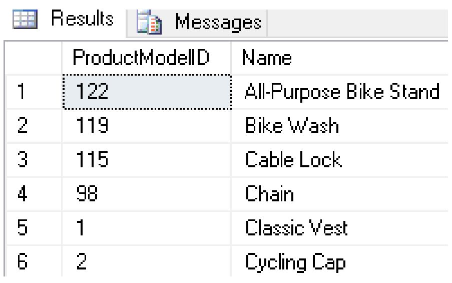
```
(2 row(s) affected)
```

```
/* INCORRECT */
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID NOT IN (3, 4)
ORDER BY Name
UNION
SELECT ProductModelID, Name
FROM dbo.Gloves;
```



```
Msg 156, Level 15, State 1, Line 386
Incorrect syntax near the keyword 'UNION'
```

```
/* CORRECT */
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID NOT IN (3, 4)
UNION
SELECT ProductModelID, Name
FROM dbo.Gloves
ORDER BY Name;
```



	ProductModelID	Name
1	122	All-Purpose Bike Stand
2	119	Bike Wash
3	115	Cable Lock
4	98	Chain
5	1	Classic Vest
6	2	Cycling Cap

- second example

```
SELECT pp.LastName, pp.FirstName, e.JobTitle
INTO dbo.EmployeeOne
FROM Person.Person AS pp JOIN HumanResources.Employee AS e
ON e.BusinessEntityID = pp.BusinessEntityID
WHERE LastName = 'Johnson';
GO
```

```
(3 row(s) affected)
```

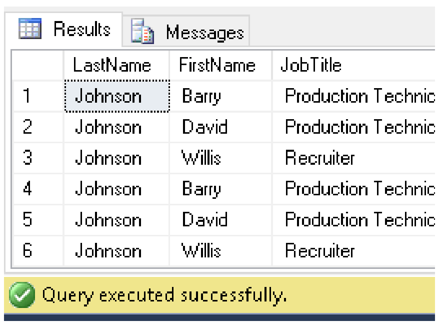
```
SELECT pp.LastName, pp.FirstName, e.JobTitle
INTO dbo.EmployeeTwo
FROM Person.Person AS pp JOIN HumanResources.Employee AS e
ON e.BusinessEntityID = pp.BusinessEntityID
WHERE LastName = 'Johnson';
GO
```

```
(3 row(s) affected)
```

```
SELECT pp.LastName, pp.FirstName, e.JobTitle
INTO dbo.EmployeeThree
FROM Person.Person AS pp JOIN HumanResources.Employee AS e
ON e.BusinessEntityID = pp.BusinessEntityID
WHERE LastName = 'Johnson';
GO
```

```
(3 row(s) affected)
```

```
SELECT LastName, FirstName, JobTitle
FROM dbo.EmployeeOne
UNION ALL
SELECT LastName, FirstName, JobTitle
FROM dbo.EmployeeTwo
UNION ALL
SELECT LastName, FirstName, JobTitle
FROM dbo.EmployeeThree;
GO
```

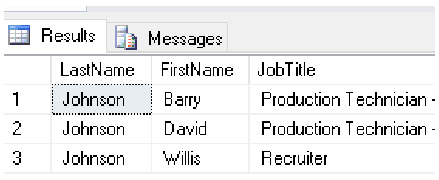


Results | Messages

	LastName	FirstName	JobTitle
1	Johnson	Barry	Production Technic
2	Johnson	David	Production Technic
3	Johnson	Willis	Recruiter
4	Johnson	Barry	Production Technic
5	Johnson	David	Production Technic
6	Johnson	Willis	Recruiter

Query executed successfully.

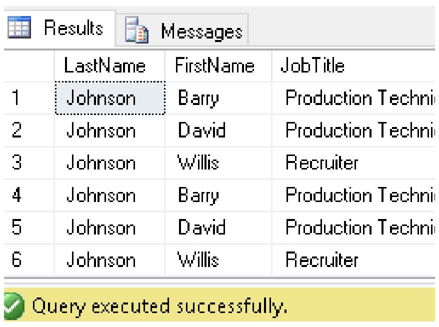
```
SELECT LastName, FirstName, JobTitle
FROM dbo.EmployeeOne
UNION
SELECT LastName, FirstName, JobTitle
FROM dbo.EmployeeTwo
UNION
SELECT LastName, FirstName, JobTitle
FROM dbo.EmployeeThree;
GO
```



Results | Messages

	LastName	FirstName	JobTitle
1	Johnson	Barry	Production Technician -
2	Johnson	David	Production Technician -
3	Johnson	Willis	Recruiter

```
SELECT LastName, FirstName,JobTitle
FROM dbo.EmployeeOne
UNION ALL
(
SELECT LastName, FirstName, JobTitle
FROM dbo.EmployeeTwo
UNION
SELECT LastName, FirstName, JobTitle
FROM dbo.EmployeeThree
);
GO
```



Results | Messages

	LastName	FirstName	JobTitle
1	Johnson	Barry	Production Techni
2	Johnson	David	Production Techni
3	Johnson	Willis	Recruiter
4	Johnson	Barry	Production Techni
5	Johnson	David	Production Techni
6	Johnson	Willis	Recruiter

Query executed successfully.

Bibliografy:
https://msdn.microsoft.com/en-us/library/ms188047.aspx
https://msdn.microsoft.com/en-us/library/ms188385.aspx
https://msdn.microsoft.com/en-us/library/ms189798.aspx
https://msdn.microsoft.com/en-us/library/ms177673.aspx
https://msdn.microsoft.com/en-us/library/ms180199.aspx