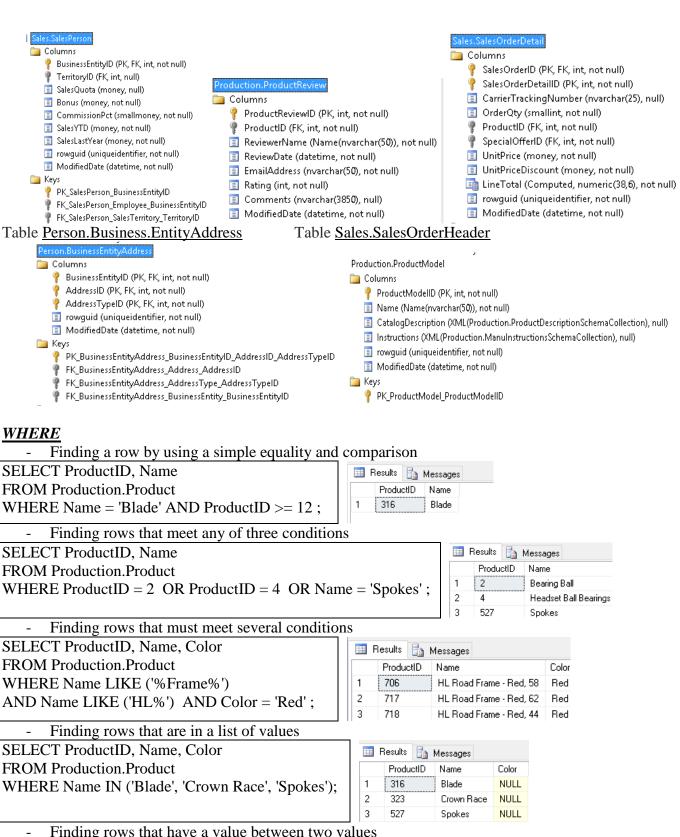
## **LABORATORY 2**

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

#### USE AdventureWorks2012 GO





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 Color 725 LL Road Frame - Red, 44 Red 2 726 LL Road Frame - Red, 48

With function

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



#### **ORDER BY**

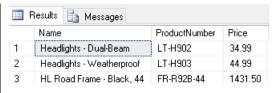
- returns all rows and only a subset of the columns (Name, ProductNumber, ListPrice)

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

- 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;



- 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;



- 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;



- returns total sales and the discounts for each product.

SELECT p.Name AS ProductName,

NonDiscountSales = (OrderQty \* UnitPrice),

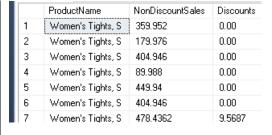
Discounts = ((OrderQty \* UnitPrice) \* UnitPriceDiscount)

FROM Production. Product AS p

INNER JOIN Sales.SalesOrderDetail AS sod

ON p.ProductID = sod.ProductID

ORDER BY ProductName DESC;



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

SELECT 'Total income is', ((OrderQty \* UnitPrice) \* (1.0 - UnitPriceDiscount)), ' for ',

FROM Production.Product AS p INNER JOIN Sales.SalesOrderDetail AS sod

ON p.ProductID = sod.ProductID

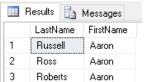
p.Name AS ProductName

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

- 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;



-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.

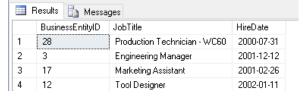
SELECT DepartmentID, Name, GroupName

FROM HumanResources.Department

ORDER BY DepartmentID OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;

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

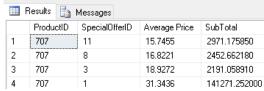
SELECT BusinessEntityID, JobTitle, HireDate FROM HumanResources. Employee ORDER BY DATEPART(year, HireDate);



## **GROUP BY**

- 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;



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

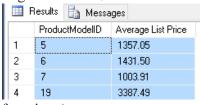
SELECT ProductModelID, AVG(ListPrice) AS [Average List Price]

FROM Production.Product

WHERE ListPrice > \$1000

GROUP BY ProductModelID

ORDER BY ProductModelID;



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

SELECT AVG(OrderOty) AS [Average Quantity],

NonDiscountSales = (OrderQty \* UnitPrice)

FROM Sales.SalesOrderDetail

GROUP BY (OrderQty \* UnitPrice)

ORDER BY (OrderQty \* UnitPrice) DESC;

III F	Results 🛅 Messag	ges
	Average Quantity	NonDiscountSales
1	26	30992.91
2	14	27607.9188
3	21	27536.0085
4	30	25514.85

- 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);

- finds the maximum/minimim price of products

SELECT ProductID, MAX(UnitPrice) AS [Maximum Price]

FROM Sales.SalesOrderDetail

**GROUP BY ProductID** 

ORDER BY MAX(UnitPrice);

	ProductID	Average Price
1	873	1.3282
2	870	2.8325
3	877	4.5343

	ProductID	Maximum Price
1	873	2.29
2	922	3.99
3	923	4.99
4	921	4.99

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

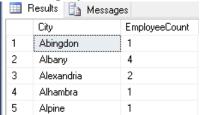


- retrieves the total for each SalesOrderID from the SalesOrderDetail table.

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

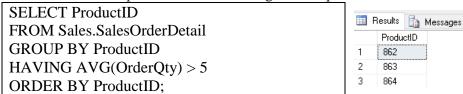
- the number of employees for each City (Address table joined to the EmployeeAddress table).

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;

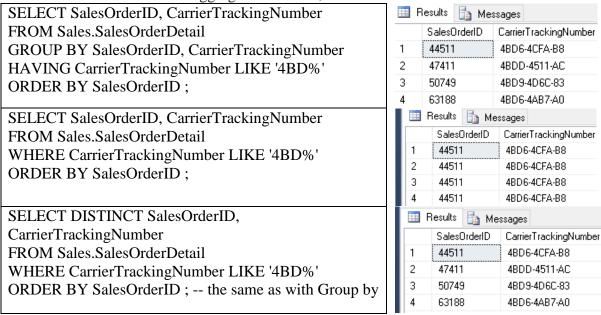


#### **HAVING**

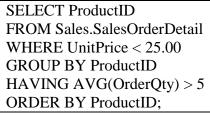
- a HAVING clause with an aggregate function. It groups the rows in the SalesOrderDetail table by product ID and eliminates products whose average order quantities are five or less.

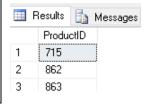


- a HAVING clause without aggregate functions, with the LIKE clause in the HAVING clause.



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





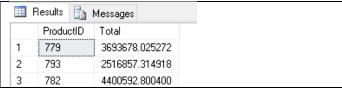
- groups the SalesOrderDetail table by product ID and includes only those groups of products that have orders totaling more than \$1000000.00 and whose average order quantities are less than 3.

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;

	Results 🚹	Messages	
	ProductID	AverageQuantity	Total
1	779	2	3693678.025272
2	793	2	2516857.314918
3	750	1	1340419.942000
4	773	2	1217210.359959

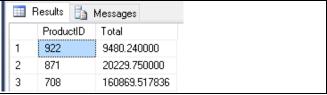
- the products that have had total sales greater than \$2000000.00

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



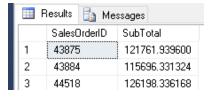
- to make sure there are at least 1500 items involved in the calculations for each product, use HAVING COUNT(\*) > 1500 to eliminate the products that return totals for fewer than 1500 items sold.

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



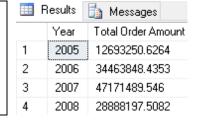
- uses a HAVING clause retrieves the total for each SalesOrderID from the SalesOrderDetail table that exceeds \$100000.00.

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



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

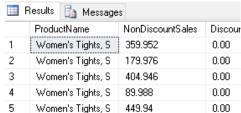
SELECT DATEPART(yyyy,OrderDate) AS N'Year', SUM(TotalDue) AS N'Total Order Amount' FROM Sales.SalesOrderHeader GROUP BY DATEPART(yyyy,OrderDate) HAVING DATEPART(yyyy,OrderDate) >= N'2003' ORDER BY DATEPART(yyyy,OrderDate);



## **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;



- 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;



- 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;

- 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;

- 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:



- cross join

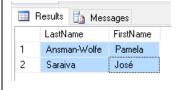
SELECT p.Name, pr.ProductReviewID
FROM Production.Product AS p
CROSS JOIN Production.ProductReview AS pr
ORDER BY ProductReviewID DESC;



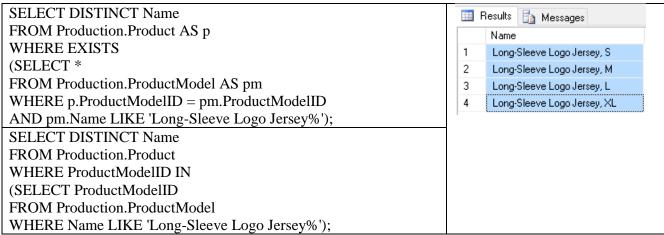
## **QUERY IN QUERY**

- retrieves one the employee for which the bonus in the SalesPerson table is 5000.00 and 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).

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);



- 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.



- uses in from

SELECT a.Name FROM (SELECT

FROM (SELECT p.Name, pr.ProductReviewID

FROM Production. Product AS p

INNER JOIN Production. ProductReview AS pr

ON p.ProductID = pr.ProductID) a

Name
Mountain Bike Socks, M
HL Mountain Pedal
HL Mountain Pedal
Road-550-W Yellow, 40
Touring-2000 Blue, 46

- 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

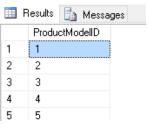
GROUP BY p1.ProductModelID

HAVING MAX(p1.ListPrice) >= ANY

(SELECT AVG(p2.ListPrice)

FROM Production. Product AS p2

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')));



## **SELECT INTO**

- creates the table NewProducts.

SELECT \* INTO dbo.NewProducts

FROM Production.Product

WHERE ListPrice > \$25 AND ListPrice < \$100;



- the result set includes the ProductModelID and Name from both 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

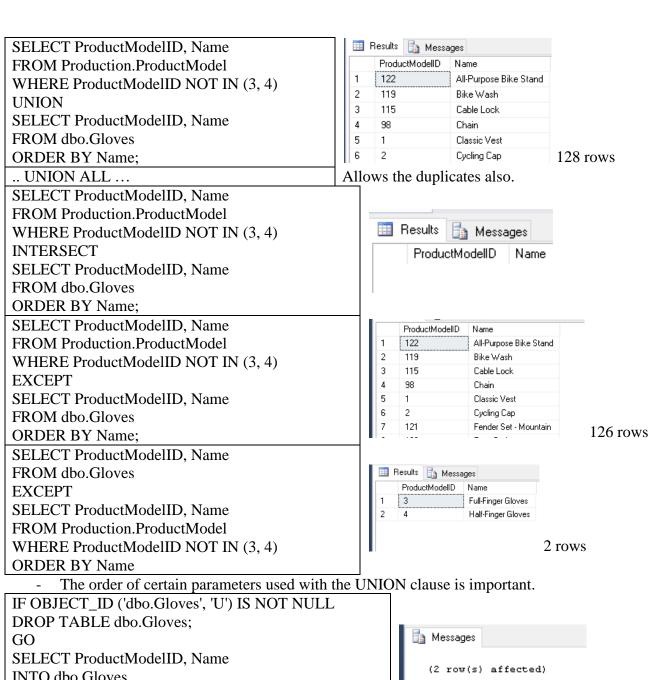
FROM Production.ProductModel

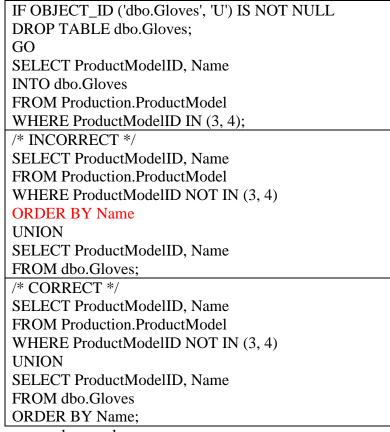
WHERE ProductModelID IN (3, 4);

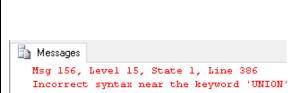
#### **UNION**

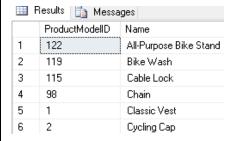
- The simple union/intersect/except.









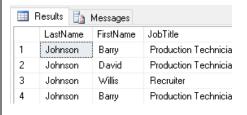


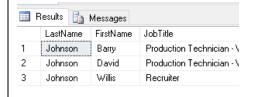
SELECT pp.LastName, pp.FirstName, e.JobTitle INTO dbo.EmployeeOne FROM Person.Person AS pp JOIN HumanResources.Employee AS e ON e.BusinessEntityID = pp.BusinessEntityID WHERE LastName = 'Johnson'; SELECT pp.LastName, pp.FirstName, e.JobTitle INTO dbo.EmployeeTwo FROM Person. Person AS pp JOIN HumanResources.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

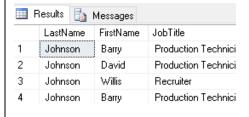
(3 row(s) affected)

(3 row(s) affected)

(3 row(s) affected)







# Bibliografy:

**UNION ALL** 

**UNION** 

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

SELECT LastName, FirstName, JobTitle

(SELECT LastName, FirstName, JobTitle

SELECT LastName, FirstName, JobTitle

FROM dbo.EmployeeOne

FROM dbo.EmployeeTwo

FROM dbo.EmployeeThree