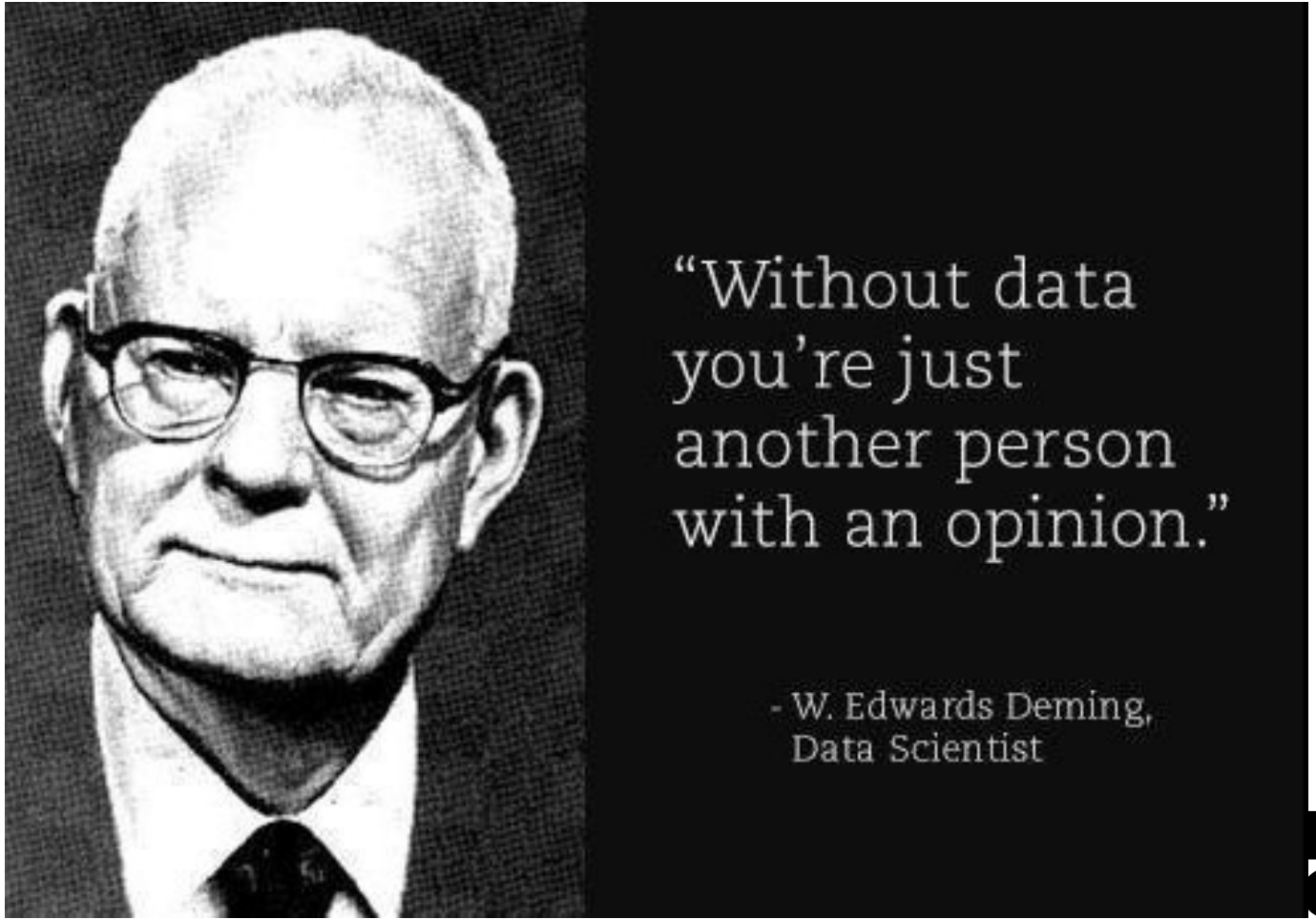
Relational Databases & Datawarehousing

SQL: basic concepts revisited


"Without data
you're just
another person
with an opinion."
- W. Edwards Deming,
Data Scientist

HO
GENT

Introduction

HO
GENT

Overview

- (Microsoft) SQL
  - Working with 1 table: SELECT, Statistical functions, GROUP BY
  - Working with > 1 tables: JOIN, UNION, subquery's, correlated subquery's
  - Modifying data: insert, update, delete
  - Views

HO GENT

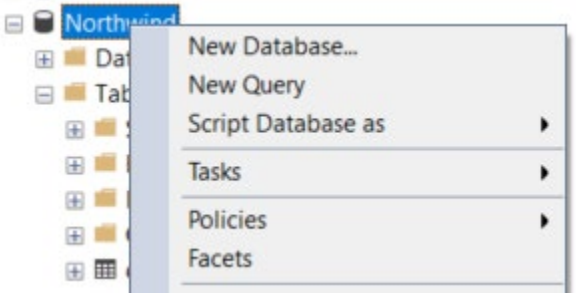
SQL Server

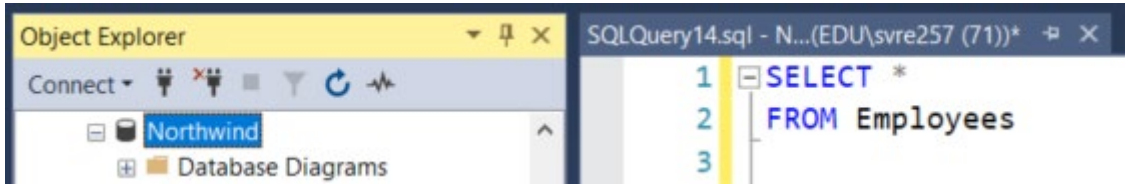
- SQL Server:
  - Management
    - Installation, configuration and security of SQL Server.
    - Database creation
    - Database management: backup, restore, ...
    - Use SQL Server Management Studio

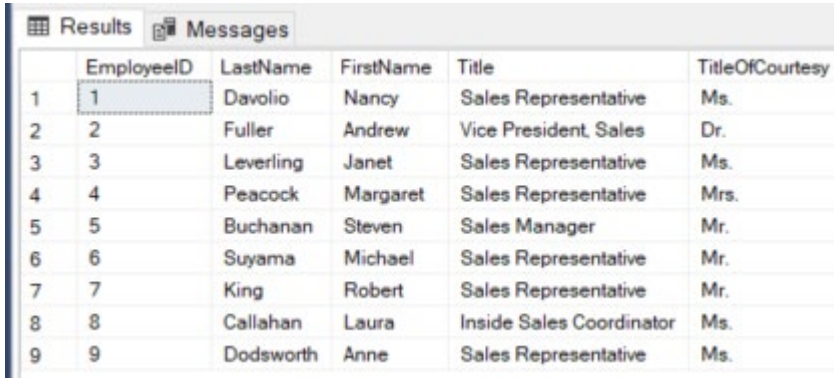
HO GENT

Writing queries

- Use SQL Server Management Studio







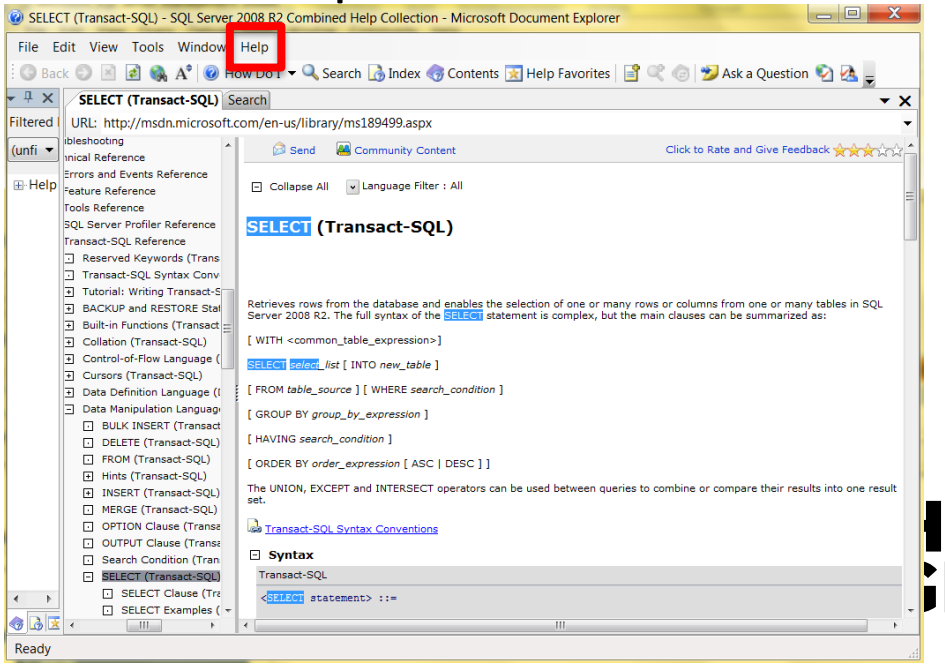
	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy
1	1	Davolio	Nancy	Sales Representative	Ms.
2	2	Fuller	Andrew	Vice President, Sales	Dr.
3	3	Leverling	Janet	Sales Representative	Ms.
4	4	Peacock	Margaret	Sales Representative	Mrs.
5	5	Buchanan	Steven	Sales Manager	Mr.
6	6	Suyama	Michael	Sales Representative	Mr.
7	7	King	Robert	Sales Representative	Mr.
8	8	Callahan	Laura	Inside Sales Coordinator	Ms.
9	9	Dodsworth	Anne	Sales Representative	Ms.

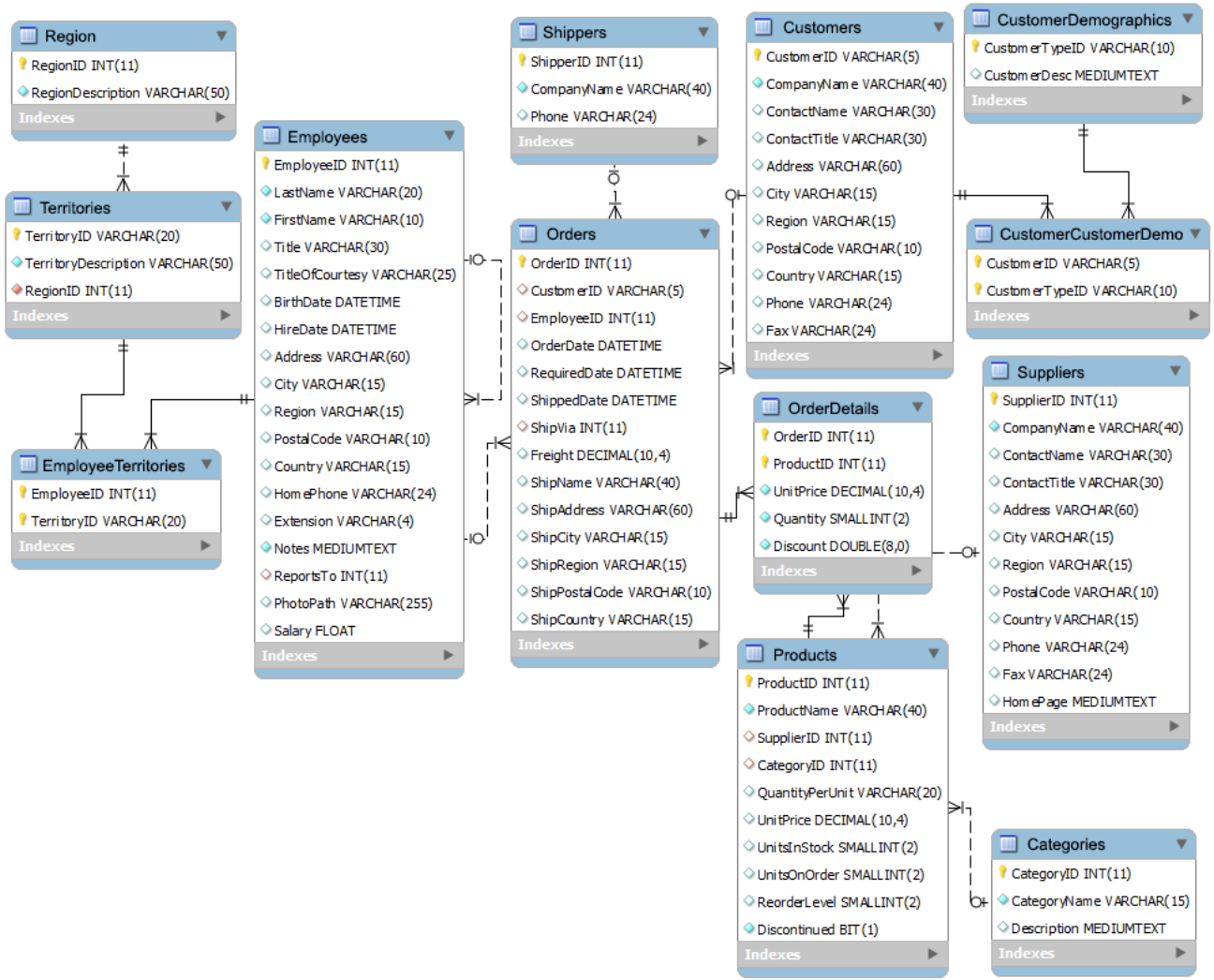
HO GENT

Help!

- The help menu offers online help about Microsoft SQL

How do I write a correct SELECT statement? Help!





‘Northwind’ DB: diagram

HO GENT

SQL - standards and dialects

- Definition
  - Relational data language for relational database systems.
  - Non procedural language
- Standards

Year	Name	Comments
1986	SQL-86	First formalized by ANSI.
1989	SQL-89	Minor revision that added integrity constraints, adopted as FIPS 127-1.
1992	**SQL-92**	**Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.**
1999	SQL:1999	Added regular expression matching, recursive queries (e.g. transitive closure), triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice versa (SQL/JRT).
2003	SQL:2003	Introduced XML-related features (SQL/XML), *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006	ISO/IEC 9075-14:2006 defines ways that SQL can be used with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database, and publishing both XML and conventional SQL-data in XML form. In addition, it lets applications integrate queries into their SQL code with XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.[40]
2008	SQL:2008	Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement.[41]
2011	SQL:2011	Adds temporal data definition and manipulation.
2016	SQL:2016	Adds row pattern matching, polymorphic table functions, JSON.

Why Microsoft SQL Server?

378 systems in ranking, September 2021

Rank			DBMS	Database Model	Score		
Sep 2021	Aug 2021	Sep 2020			Sep 2021	Aug 2021	Sep 2020
1.	1.	1.	Oracle	Relational, Multi-model	1271.55	+2.29	-97.82
2.	2.	2.	MySQL	Relational, Multi-model	1212.52	-25.69	-51.72
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	970.85	-2.50	-91.91
4.	4.	4.	PostgreSQL	Relational, Multi-model	577.50	+0.45	+35.22
5.	5.	5.	MongoDB	Document, Multi-model	496.50	-0.04	+50.02
6.	6.	↑ 7.	Redis	Key-value, Multi-model	171.94	+2.05	+20.08
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	166.56	+1.09	+5.32
8.	8.	8.	Elasticsearch	Search engine, Multi-model	160.24	+3.16	+9.74
9.	9.	9.	SQLite	Relational	128.65	-1.16	+1.98
10.	↑ 11.	10.	Cassandra	Wide column	118.99	+5.33	-0.18
11.	↓ 10.	11.	Microsoft Access	Relational	116.94	+2.10	-1.51
12.	12.	12.	MariaDB	Relational, Multi-model	100.70	+1.72	+9.09
13.	13.	13.	Splunk	Search engine	91.61	+1.01	+3.71
14.	14.	↑ 15.	Hive	Relational	85.58	+1.64	+14.41
15.	15.	↑ 17.	Microsoft Azure SQL Database	Relational, Multi-model	78.26	+3.11	+17.81

Source: db-engines.com

SQL - Overview

- SQL consists of 3 sub languages
  - Data Definition Language (DDL)
    - creation of a database, defining database objects (tables, stored procedures, views,…)
    - CREATE, ALTER, DROP
  - Data Manipulation Language (DML)
    - Querying and manipulating data in a database
    - SELECT, INSERT, UPDATE, DELETE
  - Data Control Language (DCL)
    - Data security and authorisation
    - GRANT, REVOKE, DENY

HO GENT

SQL - Overview

- Additional language elements: operators, functions, control of flow (dialects!)

HO
GENT

SELECT

HO
GENT

DML – Consulting data

- Consulting one table
  - Basic form
  - SELECT clause
  - WHERE clause
  - Row formatting
  - Statistical functions
  - Grouping
- Consulting >1 table

HO GENT

Basic form of SELECT statement

- SELECT for consulting one table

```
SELECT [ALL | DISTINCT] {*|expression [, expression ...]}
FROM table name
[WHERE conditions(s)]
[GROUP BY column name [, column name ...]
[HAVING conditions(s)]
[ORDER BY {column name |seq nr}{ASC|DESC}[,...]
```

  - SELECT clause: specifies the columns to show in the ouput. DISTINCT filters out duplicate lines
  - FROM clause: table name
  - WHERE clause : filter condition on individual lines in the output
  - GROUP BY : grouping of data
  - HAVING clause : filter condition on groups
  - ORDER BY clause : sorting

HO GENT

SELECT

- SELECT clause: specification of the columns
  - All columns from table: use *
    - SELECT *
  - Specific columns: use columns names or expression
    - SELECT column1 , column2, column3*column4, …

HO GENT

SELECT

- Example: Show all data of all products

```
SELECT *
FROM Products
```

Results | Messages

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	1	Chai	1	1	10 boxes x 20 bags	18,00	39	0	10	0
2	2	Chang	1	1	24 - 12 oz bottles	19,00	17	40	25	0
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10,00	13	70	25	0
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22,00	53	0	0	0
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21,35	0	0	0	1

HO GENT

SELECT

- Example: Show for all a products productID, name and unitprice

```
SELECT productid, productname, unitprice
FROM Products
```

	productid	productname	unitprice
1	1	Chai	18,00
2	2	Chang	19,00
3	3	Aniseed Syrup	10,00
4	4	Chef Anton's Cajun Seasoning	22,00
5	5	Chef Anton's Gumbo Mix	21,35
6	6	Grandma's Boysenberry Spread	25,00
7	7	Uncle Bob's Organic Dried Pears	30,00
8	8	Northwoods Cranberry Sauce	40,00

HO GENT

SELECT … WHERE

- WHERE clause
  - Specification of conditions for individual rows
- Example: Show productid, productname and unitprice of all products from category 1

```
SELECT productid, productname, unitprice
FROM Products
WHERE categoryID = 1
```

	productid	productname	unitprice
1	1	Chai	18,00
2	2	Chang	19,00
3	24	Guaraná Fantástica	4,50
4	34	Sasquatch Ale	14,00
5	35	Steeleye Stout	18,00
6	38	Côte de Blaye	263,50
7	39	Chartreuse verte	18,00
8	43	Ipoh Coffee	46,00
9	67	Laughing Lumberjack Lager	14,00
10	70	Outback Lager	15,00
11	75	Rhönbräu Klosterbier	7,75
12	76	Lakkalikööri	18,00

HO GENT

SELECT … WHERE

- Use of literals
  - Numeric values: ... WHERE categoryID = 1
  - Alphanumeric values: ... WHERE productName = 'Chai'
  - Dates: ... WHERE orderDate = '4/15/2018' (15th april 2018)

SELECT … WHERE

- Conditions for rows
  - Comparison operators
  - Wildcards
  - Logical operators
  - Interval of specific values
  - List of values
  - Unknown values
  - Use brackets () to overrule priority rules and enhance readability

HO GENT

SELECT … WHERE

- Comparison operators
  - =, >, >=, <, <=, <>
  - Example: Show productID, name, units in stock for all products with less than 5 units in stock

```
SELECT productid, productname, unitprice
FROM Products
WHERE UnitsInStock < 5
```

  - Example: Show productID, name, units in stock for all products for which the name starts with A

```
SELECT productid, productname, unitprice
FROM Products
WHERE productname >= 'A' AND productname < 'B'
```

HO GENT

SELECT … WHERE

- Wildcards (searching for patterns)
  - Always in combination with operator LIKE, NOT LIKE
  - Wildcard symbols:
    - % → arbitrary sequence of 0, 1 or more characters
    - _ → 1 character
    - [ ] → 1 character in a specified range
    - [^] → every character not in the specified range
  - Example: Show productID and name of the products for which the second letter is in the range a-k

```
SELECT productid, productname
FROM Products
WHERE productname LIKE '_[a-k]%'
```

HO GENT

SELECT … WHERE

- Logical operators
  - OR, AND, NOT (ascending priority)
  - Example

```
SELECT ProductID, ProductName, SupplierID, UnitPrice
FROM Products
WHERE ProductName LIKE 'T%' OR (ProductID = 46 AND UnitPrice > 16.00)
```

HO GENT

SELECT … WHERE

- Values in an interval
  - **BETWEEN, NOT BETWEEN**
  - Example: Select the products (name and unit price) for which the unit price is between 10 and 15 euro (boundaries included)

```
SELECT ProductName, UnitPrice
FROM Products
WHERE UnitPrice BETWEEN 10 AND 15
```

HO GENT

SELECT … WHERE

- List of values
  - **IN, NOT IN**
  - Example: Show ProductID, ProductName and SupplierID of the products supplied by suppliers with ID 1, 3 or 5

```
SELECT ProductID, ProductName, SupplierID
FROM Products
WHERE SupplierID in (1,3,5)
```

HO GENT

SELECT … WHERE

- Test for unknow (or empty) values
  - IS NULL, IS NOT NULL
    - NULL values occur if no value has been specified for a column when creating a record
    - A NULL is not equal to 0 (for numerical values), blank or empty string (for character values)!
    - NULL fields are considered as equal (for e.g. testing with DISTINCT)
    - If a NULL value appears in an expression the result is always NULL
  - Example: Select suppliers from an unknown region

```
SELECT CompanyName, Region
FROM Suppliers
WHERE Region IS NULL
```

HO GENT

SELECT … WHERE

- Be careful with NULL!

```
SELECT CompanyName, Region
FROM Suppliers
WHERE Region <> 'OR'
```

	CompanyName	Region
1	New Orleans Cajun Delights	LA
2	Grandma Kelly's Homestead	MI
3	Cooperativa de Quesos 'Las Cabras'	Asturias
4	Pavlova, Ltd.	Victoria
5	New England Seafood Cannery	MA
6	G'day, Mate	NSW
7	Ma Maison	Québec
8	Forêts d'érables	Québec

```
SELECT CompanyName, Region
FROM Suppliers
WHERE Region <> 'OR' OR Region IS NULL
```

	CompanyName	Region
1	Exotic Liquids	NULL
2	New Orleans Cajun Delights	LA
3	Grandma Kelly's Homestead	MI
4	Tokyo Traders	NULL
5	Cooperativa de Quesos 'Las Cabras'	Asturias
6	Mayumi's	NULL
7	Pavlova, Ltd.	Victoria
8	Specialty Biscuits, Ltd.	NULL
9	PB Knäckebröd AB	NULL
10	Refrescos Americanas LTDA	NULL

HO GENT

SELECT + formatting results

- Sorting data
- Elimination of duplicates
- Change column name in output
- Calculated output columns
- Comments
  - /* comments */
  - -- comments (rest of line is comment)

HO
GENT

SELECT … ORDER BY

- Sorting of data
  - ORDER BY clause
    - Sorting according to one or more sorting criteria
    - Each sorting criterion can be specified by either a column name, an expression or a sequence number that corresponds to the order of columns in the SELECT clause (starting from 1)
    - Sorting criteria are evaluated left to right
    - Default sort occurs in ascending order (ASC: default), if descending order is required specify DESC after the criterion
  - Example: Show an alphabetic list of product names

```
SELECT ProductName
FROM Products
ORDER BY ProductName      -- or ORDER BY 1
```

HO GENT

SELECT … ORDER BY

– Example: Show productid, name, categoryID of the products sorted by categoryID. If the category is the same products with the highest price appear first.

```
SELECT ProductID, ProductName, CategoryID, UnitPrice
FROM Products
ORDER BY CategoryID, UnitPrice DESC
```

	ProductID	ProductName	Catego...	UnitPrice
1	38	Côte de Blaye	1	263,50
2	43	Ipoh Coffee	1	46,00
3	2	Chang	1	19,00
4	1	Chai	1	18,00
5	39	Chartreuse verte	1	18,00
6	35	Steeleye Stout	1	18,00
7	76	Lakkalikööri	1	18,00
8	70	Outback Lager	1	15,00
9	67	Laughing Lumbe...	1	14,00
10	34	Sasquatch Ale	1	14,00
11	75	Rhönbräu Kloster...	1	7,75
12	24	Guaraná Fantásti...	1	4,50
13	63	Vegie-spread	2	43,90
14	8	Northwoods Cran...	2	40,00
15	61	Sirop d'érable	2	28,50

SELECT DISTINCT/ALL

- Uniqueness of rows
- DISTINCT filters out duplicates lines in the output
  - ALL (default) shows all rows, including duplicates
  - Example: Show all suppliers that supply products

```
SELECT SupplierID
FROM Products
ORDER BY SupplierID
```

```
SELECT DISTINCT SupplierID
FROM Products
ORDER BY SupplierID
```

HO GENT

Exercises

```
-- 1. Give the names of all products containing the word 'bröd' or with a name of 7
characters.


-- 2. Show the productname and the reorderlevel of all products with a level between 10 and
50 (boundaries included)
```

HO
GENT

Exercises – Solutions

```sql
-- 1. Give the names of all products containing the word 'bröd' or with a name of 7
characters.
SELECT ProductName
FROM Products
WHERE ProductName LIKE '%bröd%' or ProductName LIKE '_______'

-- 2. Show the productname and the reorderlevel of all products with a level between 10 and
50 (boundaries included)
SELECT ProductName, ReorderLevel
FROM Products
WHERE ReorderLevel BETWEEN 10 AND 50
```

HO GENT

SELECT and aliases

- Column names in output
  - Default : column title = name of column in table; calculated columns are unnamed
  - The AS keyword allows you to give a column a new title
    - Remark: the new column name can only be used in ORDER BY (not in WHERE, HAVING, GROUP BY)
  - Example: Select ProductID, ProductName of the products.

```
SELECT ProductID AS ProductNummer, ProductName AS 'Name Product'
FROM Products
```

HO GENT

SELECT with calculated results

- Calculated result columns
  - Arithmetic operators : +, -, /, *
  - Example: Give name and inventory value of the products

```
SELECT ProductName, UnitPrice * UnitsInStock  AS InventoryValue
FROM Products
```

	ProductName	InventoryValue
1	Chai	702,00
2	Chang	323,00
3	Aniseed Syrup	130,00
4	Chef Anton's Cajun Seasoning	1166,00
5	Chef Anton's Gumbo Mix	0,00
6	Grandma's Boysenberry Spread	3000,00

HO GENT

SELECT and use of functions

- Functies
  - String functions: left, right, len, ltrim, rtrim, substring, replace, ...
  - DateTime functions: DateAdd, DateDiff, DatePart, Day, Month, Year
    - GETDATE(): returns current date and time in DATETIME format specified by MS-SQL Server.
  - Arithmetic functions: round, floor, ceiling, cos, sin, ...
  - Aggregate functions: AVG, SUM, ...
  - ISNULL: replaces NULL values with specified value
  - Reference document: http://msdn.microsoft.com/en-us/library/ms174318.aspx

```
SELECT ISNULL(UnitPrice, 10.00)
FROM Products
```

HO GENT

SELECT and data type conversion

- Implicit conversions
  - Sometimes possible
  - Example: UnitsInStock * 0.5
    UnitInStock (int) is automatically converted to decimal

SELECT and data type conversion

- Explicit conversions
  - CAST (<value expression> AS <data type>)
  - Example: PRINT CAST(-25.25 AS INTEGER) -> -25
  - CONVERT (<data type, <expression> [, <style>])

```
SELECT CONVERT(VARCHAR, getdate(), 106) As Today
```

	Today
1	08 Feb 2021

  - FORMAT

```
SELECT *
FROM Orders
WHERE FORMAT(ShippedDate,'dd/MM/yyyy')='10/07/2020'
```

HO GENT

String functions

	SQL SERVER
concatenate	`SELECT CONCAT(Address,' ',City) FROM Employees` `SELECT Address + ' ' + City FROM Employees`
substring	`SELECT SUBSTRING(Address, 1, 5) FROM Employees`
left part	`SELECT LEFT(Address,5) FROM Employees`
right part	`SELECT RIGHT(Address,5) FROM Employees`
length	`SELECT LEN(Address) FROM Employees`
lowercase	`SELECT LOWER(Address) FROM Employees`
uppercase	`SELECT UPPER(Address) FROM Employees`
remove spaces left and right	`SELECT RTRIM(LTRIM(Address)) FROM Employees`

HO GENT

Date / time functions

	SQL SERVER
System date	`SELECT GETDATE()`

Add years, months, days to date	`SELECT DATEADD (year, 2, GETDATE())`
	`SELECT DATEADD (month, 2, GETDATE())`
	`SELECT DATEADD (day, 2, GETDATE())`

Number of years, months, days between 2 dates	`SELECT DATEDIFF(day,BIRTHDATE,GETDATE()) As NumberOfDays` `FROM Employees`
Day of the month	`SELECT DAY(GETDATE())`
Month of the year	`SELECT MONTH(GETDATE())`
Year	`SELECT YEAR(GETDATE())`

HO GENT

Date / time: examples

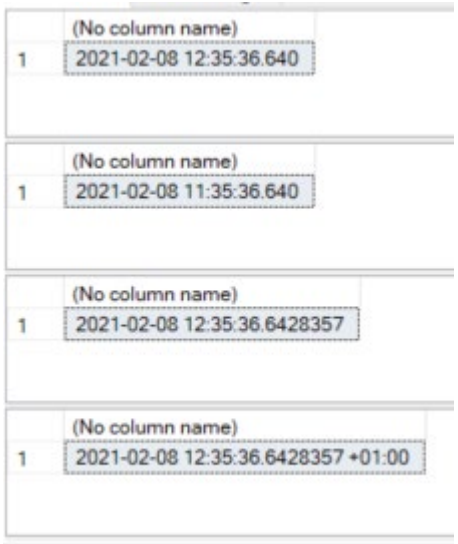
- https://msdn.microsoft.com/en-us/library/ms186724.aspx

```
SELECT GETDATE()

SELECT GETUTCDATE()

SELECT SYSDATETIME()

SELECT SYSDATETIMEOFFSET()
```



HO GENT

Arithmetic functions

	SQL SERVER
Absolute value	`SELECT ABS(-10) -- 10`
Round to give number of decimals	`SELECT ROUND(10.75, 1)  -- 10.8`
Largest integer thas is lower	`SELECT FLOOR(10.75)   -- 10`
Smallest integer that is higher	`SELECT CEILING(10.75) -- 11`

HO GENT

The case function

- Simple CASE expression:

```
SELECT City, Region,
CASE region
  WHEN 'OR' THEN 'West'
  WHEN 'MI' THEN 'North'
  ELSE 'Elsewhere'
END As RegionElaborated
FROM Suppliers
```

HO GENT

The case function

- Searched CASE expression:

```
SELECT CONVERT(varchar(20), ProductName) As 'Shortened ProductName',
   CASE
      WHEN UnitPrice IS NULL THEN 'Not yet priced'
      WHEN UnitPrice < 10 THEN 'Very Reasonable Price'
      WHEN UnitPrice >= 10 and UnitPrice < 20 THEN 'Affordable'
      ELSE 'Expensive!'
   END AS 'Price Category'
FROM Products
ORDER BY UnitPrice
```

	Shortened ProductName	Price Category
1	Geitost	Very Reasonable Price
2	Guaraná Fantástica	Very Reasonable Price
3	Konbu	Very Reasonable Price
4	Filo Mix	Very Reasonable Price
5	Tourtière	Very Reasonable Price
6	Rhönbräu Klosterbier	Very Reasonable Price
7	Tunnbröd	Very Reasonable Price
8	Teatime Chocolate Bi	Very Reasonable Price
9	Zaanse koeken	Very Reasonable Price
10	Rogede sild	Very Reasonable Price
11	Jack's New England C	Very Reasonable Price
12	Sir Rodney's Scones	Affordable
13	Aniseed Syrup	Affordable
14	Longlife Tofu	Affordable
15	Spegesild	Affordable

SELECT and strings

- String operator: concatenate

```
SELECT STR(ProductID) + ',' + ProductName AS Product
FROM Products
```

	Product
1	17,Alice Mutton
2	3,Aniseed Syrup
3	40,Boston Crab Meat

- Use of literal text (literals)

```
SELECT ProductName, '$' As Currency, Unitprice
FROM Products
```

	ProductName	Currency	Unitprice
1	Chai	$	18,00
2	Chang	$	19,00
3	Aniseed Syrup	$	10,00

HO GENT

GROUP BY and statistical functions

HO GENT

Statistical functions

- Statistical functions (aka aggregate functions)
  - SQL has 5 standard functions
    - SUM(expression): sum
    - AVG(expression): average
    - MIN(expression): minimum
    - MAX(expression): maximum
    - COUNT(*|[DISTINCT] column name): count
  - These functions give one answer per column (or group: see further) and can never be used in a where-clause

sum and average

- SUM
  - Returns the sum of all (numeric) values in a column
  - Can only be used with numeric columns
  - Example: Give the total stock value

```
SELECT SUM(UnitsInStock * UnitPrice) as InventoryValue
FROM Products
```

HO GENT

sum and average

- AVG
  - Returns the average of NOT NULL numeric values in a columns
  - Can only be used with numeric columns
  - Example: What is the average number of products in stock?

```
SELECT AVG(UnitsInStock) AS AverageStock
FROM Products
```

HO GENT

Count the number of rows

- COUNT
  - Returns the number of rows, or a number of NOT NULL values in a column
    - COUNT(*) – counts the number of rows in a SELECT
    - Example: Count the number of products (= all rows)

```
SELECT COUNT(*) as NumberOfProducts
FROM Products
```

    - COUNT (column name) – counts the number of not empty fields in a column
    - Example: Count the number of NOT NULL values in column CategoryID

```
SELECT COUNT(CategoryID) as NumberOfCategoryID
FROM Products
```

	NumberOfCategoryID
1	77

Count the number of rows

- COUNT
  - Returns the number of rows, or a number of NOT NULL values in a column
    - COUNT(DISTINCT column name) - count the number of different NOT NULL values in column producttypeid
    - Example: Count the number of different NOT NULL values in column CategoryID

```
SELECT COUNT(DISTINCT CategoryID) as NumberOfCategoryID
FROM Products
```

	NumberOfCategoryID
1	8

HO GENT

minimum and maximum

- MIN and MAX
  - Returns the smallest and largest value in a column
  - Applicable for both numeric, alphanumeric and datetime fields
  - Example: What is the cheapest and most expensive unit price?

```
SELECT MIN(UnitPrice) AS Minimum, MAX(UnitPrice) AS Maximum
FROM Products
```

HO GENT

Statistical functions - Remark

- Since a statistical function returns only **1 result**, either all expressions in the SELECT clause have to contain a statistical function, or none!
  This is slightly different if you use group by (see further).
- Statistical functions do not take into account **NULL values**.
  Exception : COUNT(*) also counts rows with NULL values.

Transact-SQL dialect

- Some statistical functions only exists in MS Transact-SQL
  - STDEV: standard deviation of column values
  - VAR: variance of column values
  - TOP:
    - Example: Select the top 5 of the cheapest products

```
SELECT TOP 5 ProductID, UnitPrice
FROM Products
ORDER BY UnitPrice
```

    - Example: Select the 5 most expensive products

```
SELECT TOP 5 ProductID, UnitPrice
FROM Products
ORDER BY UnitPrice DESC
```

Grouping with GROUP BY

- Grouping – Statistical functions per group.
  - GROUP BY clause :
    - The table is divided into groups of rows with common characteristics.
    - Per group one unique row!
    - For each group statistical functions can be applied.
    - The column names (or grouping criteria) mentioned in the GROUP BY clause can also appear in the SELECT clause

Grouping with GROUP BY

- Some examples
  - Show the number of products per category

```
SELECT CategoryID, COUNT(ProductID) As NumberOfProductsPerCategory
FROM Products
GROUP BY CategoryID
```

	CategoryID	NumberOfPr...
1	1	12
2	2	12
3	3	13
4	4	10
5	5	7
6	6	6
7	7	5
8	8	12

  - Show per category the number of products with UnitPrice > 15

```
SELECT CategoryID, COUNT(ProductID) As NumberOfProductsPerCategory
FROM Products
WHERE UnitPrice > 15
GROUP BY CategoryID
```

	CategoryID	NumberOfProdu
1	1	7
2	2	10
3	3	7
4	4	8
5	5	4
6	6	5
7	7	4
8	8	6

Filter on groups with HAVING

- HAVING clause
  - Select or reject groups based on group characteristics
  - Some examples:
    - Show the categories that contain more than 10 products

```
SELECT CategoryID, COUNT(ProductID) As NumberOfProductsPerCategory
FROM Products
GROUP BY CategoryID
HAVING COUNT(ProductID) > 10
```

	CategoryID	NumberOfProduct
1	1	12
2	2	12
3	3	13
4	8	12

    - Show the categories that contain more than 10 products with UnitPrice > 15

```
SELECT CategoryID, COUNT(ProductID) As NumberOfProductsPerCategory
FROM Products
WHERE UnitPrice > 10
GROUP BY CategoryID
HAVING COUNT(ProductID) > 10
```

	CategoryID	NumberOfProductsPerCategory
1	2	11

HO GENT

WHERE vs HAVING

- Remarks
  - WHERE vs HAVING
    - WHERE – works on individual rows
    - HAVING – works on groups / conditions on aggregation functions
  - Statistical functions can only be used in SELECT, HAVING, ORDER BY - not in WHERE, GROUP BY
  - If statistical functions appear in the SELECT, then all items in the SELECT-list have to be either statistical functions or group identifications

```
SELECT CategoryID, MIN(UnitPrice) As Minimum
FROM Products
```

```
Msg 8120, Level 16, State 1, Line 1
Column 'Products.CategoryID' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
```

HO GENT

Exercises

```
-- 1. Count the amount of products (columnname 'amount of products'), AND the amount of products in stock (=
unitsinstock not empty) (columnname 'Units in stock')
-- 2. How many employees have a function of Sales Representative (columnname 'Number of Sales
Representative')?
-- 3. Give the date of birth of the youngest employee (columnname 'Birthdate youngest') and the oldest
(columnname 'Birthdate oldest').
-- 4. What's the number of employees who will retire (at 65) within the first 20 years?
-- 5. Show a list of different countries where 2 of more suppliers are from. Order alphabeticaly.
-- 6. Which suppliers offer at least 5 products with a price less than 100 dollar? Show supplierId and the
number of different products.
-- The supplier with the highest number of products comes first.
```

HO GENT

Working with more than 1 table: JOIN

Consult > 1 table

- JOIN
  - Inner join
  - Outer join
  - Cross join
- UNION
- Subquery's
  - Simple nested query's
  - Correlated subquery's
  - Operator EXISTS
- Set Operators
- Common Table Expressions

HO GENT

JOIN

- Select columns from several tables
  - JOIN keyword : specifies which tables have to be joined and how
    - Inner join
    - Outer join
    - Cross join
    - ON keyword : specifies the JOIN condition
  - Produces 1 result set, joining the rows of both tables
  - Basic form (ANSI JOIN (SQL-92) <-> Old style join)

```
SELECT expression
FROM table1 JOIN table2 ON condition
           [JOIN table2 ON condition...]
```

```
SELECT expression
FROM table1, table2 [, table3...]
WHERE condition(s)
```

HO GENT

INNER JOIN

- Joins rows from one table with rows from another table based on common criteria in the corresponding tables.
- The relation between the fields in the corresponding tables is expressed through:
  - = (equi-join)
  - <
  - >
  - <>
  - >=
  - <=

HO GENT

INNER JOIN

- Example of equi-join
  - Give the productID, productName and CategoryName for each product
    - ANSI JOIN (SQL-92)

```
SELECT ProductID, ProductName, CategoryName
FROM Products JOIN Categories
ON Products.CategoryID = Categories.CategoryID
```

    - OR "old style join"

```
SELECT ProductID, ProductName, CategoryName
FROM Products, Categories
WHERE Products.CategoryID = Categories.CategoryID
```

HO GENT

Aliases

- USE tables aliasses (via 'AS' or blank)
  - SQL-92

```
SELECT ProductID, ProductName, CategoryName
FROM Products p JOIN Categories c
ON p.CategoryID = c.CategoryID
```

  - OR "old style join"

```
SELECT ProductID, ProductName, CategoryName
FROM Products p, Categories c
WHERE p.CategoryID = c.CategoryID
```

HO GENT

Remarks

- If the same column name is used in several tables in a query, then each column name has to be preceeded by the table name or its alias.
- Inner joins only return rows that meet the ON condition.
- If you omit (forget) the where clause in the old style join all combinations are returned
  = CROSS JOIN (= carthesian product) (see further)

INNER JOIN of > 2 tables

- JOIN of more than 2 tables
  - Example: Give for each product the ProductName, the CategoryName and the CompanyName of the supplier
  - SQL-92 :

```
SELECT p.ProductID, p.ProductName, c.CategoryName, s.CompanyName
FROM Products p JOIN Categories c ON p.CategoryID = c.CategoryID
JOIN Suppliers s ON p.SupplierID = s.SupplierID
```

  - Old style join

```
SELECT p.ProductID, p.ProductName, c.CategoryName, s.CompanyName
FROM Products p, Categories c, Suppliers s
WHERE p.CategoryID = c.CategoryID AND p.SupplierID = s.SupplierID
```

HO GENT

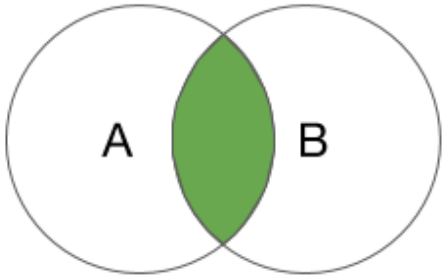
INNER JOIN of a table with itself

- Example: Show all employees and the name of whom they have to report to

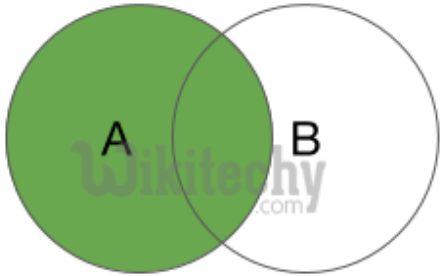
```
SELECT e1.EmployeeID, e1.Firstname + ' ' + e1.LastName As Employee,
e2.Firstname + ' ' + e2.LastName As ReportsTo
FROM Employees e1 JOIN Employees e2
ON e1.ReportsTo = e2.EmployeeID
```

OUTER JOIN

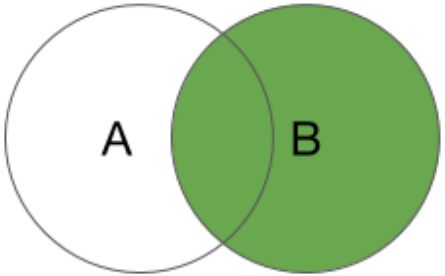
- Returns all records from 1 table, even if there is no corresponding record in the other table
- 3 types of an outer join
  - LEFT OUTER JOIN
    - Returns all rows of the first table in the FROM clause(SQL-92)
  - RIGHT OUTER JOIN
    - Returns all rows of the second table in the FROM clause(SQL-92)
  - FULL OUTER JOIN
    - Returns all rows of the first and the second table in the FROM clause(SQL-92) even if there is no corresponding record in the other table



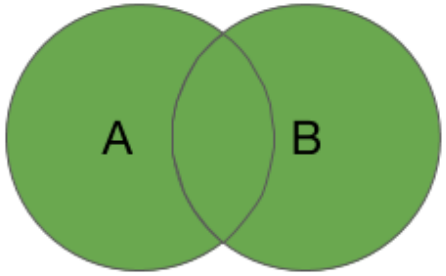
INNER JOIN



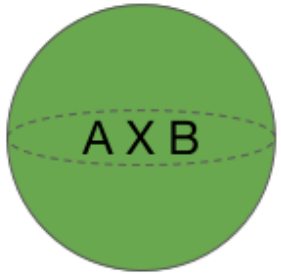
LEFT OUTER JOIN



RIGHT OUTER JOIN



FULL OUTER JOIN



CARTESIAN (CROSS) JOIN

LEFT OUTER JOIN

- Example: Show the number of shippings per Shipper

```
SELECT s.ShipperID, s.CompanyName, COUNT(OrderID) As NumberOfShippings
FROM Shippers s JOIN Orders o
ON s.shipperID = o.shipVia
GROUP BY s.ShipperID, s.CompanyName
```

	ShipperID	CompanyName	NumberOfShippings
1	1	Speedy Express	249
2	2	United Package	326
3	3	Federal Shipping	255

```
SELECT s.ShipperID, s.CompanyName, COUNT(OrderID) As NumberOfShippings
FROM Shippers s LEFT JOIN Orders o
ON s.shipperID = o.shipVia
GROUP BY s.ShipperID, s.CompanyName
```

	ShipperID	CompanyName	NumberOfShippings
1	1	Speedy Express	249
2	2	United Package	326
3	3	Federal Shipping	255
4	4	Total Shipping	0
5	5	Federal Express	0

HO GENT

RIGHT OUTER JOIN

- Example: Give the employees to whom no one reports

```
SELECT e1.Firstname + ' ' + e1.LastName As Employee,
e2.Firstname + ' ' + e2.LastName As ReportsTo
FROM Employees e1 RIGHT JOIN Employees e2
ON e1.ReportsTo = e2.EmployeeID
WHERE e1.Firstname + ' ' + e1.LastName IS NULL
```

	Employee	ReportsTo
1	NULL	Nancy Davolio
2	NULL	Janet Leverling
3	NULL	Margaret Peacock
4	NULL	Michael Suyama
5	NULL	Robert King
6	NULL	Laura Callahan
7	NULL	Anne Dodsworth

HO GENT

FULL OUTER JOIN

- FULL OUTER JOIN is the combination (=UNION) of
  LEFT and RIGHT OUTER JOIN

```
SELECT o.OrderID, s.ShipperID, s.CompanyName
FROM Shippers s FULL OUTER JOIN Orders o
ON s.shipperID = o.shipVia
```

	OrderID	ShipperID	CompanyName
806	10975	3	Federal Shippi...
807	10977	3	Federal Shippi...
808	10984	3	Federal Shippi...
809	10990	3	Federal Shippi...
810	10992	3	Federal Shippi...
811	10993	3	Federal Shippi...
812	10994	3	Federal Shippi...
813	10995	3	Federal Shippi...
814	11000	3	Federal Shippi...
815	11003	3	Federal Shippi...
816	11008	3	Federal Shippi...
817	11012	3	Federal Shippi...
818	11014	3	Federal Shippi...
819	11019	3	Federal Shippi...
820	11025	3	Federal Shippi...
821	11032	3	Federal Shippi...
822	11033	3	Federal Shippi...
823	11036	3	Federal Shippi...
824	11040	3	Federal Shippi...
825	11047	3	Federal Shippi...
826	11048	3	Federal Shippi...
827	11051	3	Federal Shippi...
828	11057	3	Federal Shippi...
829	11058	3	Federal Shippi...
830	11061	3	Federal Shippi...
831	NULL	4	Total Shipping
832	NULL	5	Federal Express

CROSS JOIN

- In a cross join the number of rows in the result set equals the number of rows in the first table multiplied by the number of rows in the second table
- Application: Generate all combinations
- Example: Make a schedule in which each employee should contact each customer

```
SELECT e.EmployeeID, e.FirstName + ' ' + e.LastName, e.Title,
c.CompanyName, c.ContactName, c.ContactTitle, c.Phone
FROM Employees e CROSS JOIN Customers c
```

	EmployeeID	(No column name)	Title	CompanyName	ContactName	ContactTitle	Phone
1	1	Nancy Davolio	Sales Representative	Alfreds Futterkiste	Maria Anders	Sales Representative	030-0074321
2	1	Nancy Davolio	Sales Representative	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	(5) 555-4729

SET OPERATORS: UNION – INTERSECT - EXCEPT

HO GENT

UNION

- A UNION combines the result of 2 or more queries
  - Basic form

```
SELECT ...  FROM ... WHERE ...
UNION
SELECT ...  FROM ... WHERE ...
ORDER BY ...
```

  - Rules
    - Both SELECTs have to contain an equal number of columns
    - Corresponding columns from both SELECTs should have compatible data types
    - The columns names or aliases from the first SELECT or shown
    - The result set does not contain duplicates. To keep duplicates use UNION ALL
    - At the end an ORDER BY can be added.
      Column names or expressions can't be used in the ORDER BY if they differ between the two SELECTs. In this case use column numbers for sorting.

UNION

- Example: Give an overview of all employees (lastname and firstname, city and postal code) and all customers (name, city and postal code)

```
SELECT LastName  + ' ' + FirstName as Name, City, Postalcode
FROM Employees
UNION
SELECT CompanyName, City, Postalcode
FROM Customers
```

	Name	City	Postalcode
1	Alfreds Futterkiste	Berlin	12209
2	Ana Trujillo Emparedados y helados	México D.F.	05021
3	Antonio Moreno Taquería	México D.F.	05023
4	Around the Horn	London	WA1 1DP
5	Berglunds snabbköp	Luleå	S-958 22
6	Blauer See Delikatessen	Mannheim	68306

HO GENT

INTERSECT

- Which records are in the intersection?

```
SELECT City, Country FROM Customers
INTERSECT
SELECT City, Country FROM Suppliers
```

	City	Country
1	Berlin	Germany
2	London	UK
3	Montréal	Canada
4	Paris	France
5	Sao Paulo	Brazil

HO GENT

EXCEPT

- The EXCEPT operator subtracts a result set from another result set.
  - Example: Which products have never been ordered?

```
SELECT ProductID
FROM Products
EXCEPT
SELECT ProductID
FROM OrderDetails
```

Exercises

```
-- 1. Which suppliers (SupplierID and CompanyName) deliver Dairy Products?
-- 2. Give for each supplier the number of orders that contain products of that supplier.
-- Show supplierID, companyname and the number of orders.
-- Order by companyname.
-- 3. What's for each category the lowest UnitPrice? Show category name and unit price.
-- 4. Give for each ordered product: productname, the least (columnname 'Min amount ordered') and the most
ordered (columnname 'Max amount ordered'). Order by productname.
-- 5. Give a summary for each employee with orderID, employeeID and employeename.
-- Make sure that the list also contains employees who don't have orders yet.
```

HO
GENT

Exercises – Solutions

```sql
-- 1. Which suppliers (SupplierID and CompanyName) deliver Dairy Products?
SELECT DISTINCT s.SupplierID, s.CompanyName
FROM Suppliers s JOIN Products p ON s.SupplierID = p.SupplierID
JOIN Categories c ON p.CategoryID = c.CategoryID
WHERE c.CategoryName LIKE '%Dairy%'

-- 2. Give for each supplier the number of orders that contain products of that supplier.
-- Show supplierID, companyname and the number of orders.
-- Order by companyname.
select s.SupplierID, s.CompanyName, count(DISTINCT od.OrderID) As NrOfOrders
from Suppliers s join Products p ON s.SupplierID = p.SupplierID
JOIN OrderDetails od ON od.ProductID = p.ProductID
GROUP BY s.SupplierID, s.CompanyName
ORDER BY s.CompanyName

-- 3. What's for each category the lowest UnitPrice? Show category name and unit price.
SELECT c.CategoryName, MIN(p.UnitPrice) As 'Minimum UnitPrice'
FROM Products p join Categories c ON p.CategoryID = c.CategoryID
GROUP BY c.CategoryName
```

HO GENT

Exercises – Solutions

```sql
-- 4. Give for each ordered product: productname, the least (columnname 'Min amount ordered') and the most
ordered (columnname 'Max amount ordered'). Order by productname.
SELECT p.ProductName, MIN(od.Quantity) As 'Min amount ordered', Max(od.Quantity) As 'Max amount ordered'
FROM Products p join OrderDetails od ON p.ProductID = od.ProductID
GROUP BY p.ProductName
ORDER BY p.ProductName

-- 5. Give a summary for each employee with orderID, employeeID and employeename.
-- Make sure that the list also contains employees who don't have orders yet.
SELECT e.EmployeeID, e.FirstName + ' ' + e.LastName As 'Name', o.OrderID
FROM Employees e left join Orders o on e.EmployeeID = o.EmployeeID
```

HO GENT