SQL Practicum

Michael Rose

```
# rmarkdown::render('sql_practicum.Rmd', params = 'ask')
sql_con <- dbConnect(odbc(), Driver = "ODBC Driver 17 for SQL Server",
    Server = "localhost", Database = "Northwind_SPP", UID = "SA",
    PWD = rstudioapi::askForPassword("Database password"), Port = 1433)
```

Introduction

This is an in-depth analysis of a production size database using the **MS SQL Server** dialect of SQL.

The goal is to complete a series of queries that make use of a wide range of the functionality built within sql.

For the sake of readability, I have limited the output to the top 25 rows. If a query has more than 25 rows, it will be noted above the output.

If you wish to see the full output, in the code there is code chunk options. Remove max.print = 25 for full output.

Overview

Why was this project chosen?

SQL is a very extensible language. Built off a few primitives (select, from, where), the combination of different functions provides a rich toolset for exploring databases. I wanted to build familiarity with some of the features available through practice.

What does your project do / how does it work?

The following project consists of 50 sql queries, ranging in difficulty from simple and moving on top advanced. Each section builds on knowledge built in the last.

What tools did you use?

I used Rmarkdown for the pdf formatting, R language for connecting to the database, microsoft sql server 2017 for setting a database on my localhost, azure data studio for managing permissions for connections, and odbc for connection drivers. The queries themselves are in MS SQL. The database was provided by **Sylvia Moestl Vasilik** in accompaniment with her book **SQL Practice Problems**.

What did you learn?

I learned quite a bit of SQL. Thanks to these challenging exercises, I feel much more comfortable with the language and its functionality. I also got to explore the effects of a variety of functions and methods for querying data.

What is the format of the data?

The data is stored in a MS SQL Server. It contains a variety of tables which are shown in an upcoming section.

What other things would you have liked to do with this project?

I would have liked to make some models, but unfortunately this data does not contain any fields worth predicting on.

SQL Analysis

0.1 | Tables

```
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
```

Table 1: 9 records

TABLE_NAME
Categories
CustomerGroupThresholds
Customers
Employees
OrderDetails
Orders
Products
Shippers
Suppliers

0.2 | Table Information

```
SELECT TABLE_NAME,
       COLUMN_NAME,
       DATA_TYPE
FROM   INFORMATION_SCHEMA.COLUMNS
ORDER BY TABLE_NAME
```

Table 2: 78 records

TABLE_NAME	COLUMN_NAME	DATA_TYPE
Categories	CategoryID	int
Categories	CategoryName	nvarchar
Categories	Description	ntext
CustomerGroupThresholds	CustomerGroupName	varchar
CustomerGroupThresholds	RangeBottom	money
CustomerGroupThresholds	RangeTop	money
Customers	CustomerID	nchar

TABLE_NAME	COLUMN_NAME	DATA_TYPE
Customers	CompanyName	nvarchar
Customers	ContactName	nvarchar
Customers	ContactTitle	nvarchar
Customers	Address	nvarchar
Customers	City	nvarchar
Customers	Region	nvarchar
Customers	PostalCode	nvarchar
Customers	Country	nvarchar
Customers	Phone	nvarchar
Customers	Fax	nvarchar
Employees	EmployeeID	int
Employees	LastName	nvarchar
Employees	FirstName	nvarchar
Employees	Title	nvarchar
Employees	TitleOfCourtesy	nvarchar
Employees	BirthDate	datetime
Employees	HireDate	datetime
Employees	Address	nvarchar
Employees	City	nvarchar
Employees	Region	nvarchar
Employees	PostalCode	nvarchar
Employees	Country	nvarchar
Employees	HomePhone	nvarchar
Employees	Extension	nvarchar
Employees	Notes	ntext
Employees	ReportsTo	int
Employees	PhotoPath	nvarchar
OrderDetails	OrderID	int
OrderDetails	ProductID	int
OrderDetails	UnitPrice	money
OrderDetails	Quantity	smallint
OrderDetails	Discount	real
Orders	OrderID	int
Orders	CustomerID	nchar
Orders	EmployeeID	int
Orders	OrderDate	datetime
Orders	RequiredDate	datetime
Orders	ShippedDate	datetime
Orders	ShipVia	int
Orders	Freight	money
Orders	ShipName	nvarchar
Orders	ShipAddress	nvarchar
Orders	ShipCity	nvarchar
Orders	ShipRegion	nvarchar
Orders	ShipPostalCode	nvarchar
Orders	ShipCountry	nvarchar
Products	ProductID	int
Products	ProductName	nvarchar
Products	SupplierID	int
Products	CategoryID	int
Products	QuantityPerUnit	nvarchar
Products	UnitPrice	money

TABLE_NAME	COLUMN_NAME	DATA_TYPE
Products	UnitsInStock	smallint
Products	UnitsOnOrder	smallint
Products	ReorderLevel	smallint
Products	Discontinued	bit
Shippers	ShipperID	int
Shippers	CompanyName	nvarchar
Shippers	Phone	nvarchar
Suppliers	SupplierID	int
Suppliers	CompanyName	nvarchar
Suppliers	ContactName	nvarchar
Suppliers	ContactTitle	nvarchar
Suppliers	Address	nvarchar
Suppliers	City	nvarchar
Suppliers	Region	nvarchar
Suppliers	PostalCode	nvarchar
Suppliers	Country	nvarchar
Suppliers	Phone	nvarchar
Suppliers	Fax	nvarchar
Suppliers	HomePage	ntext

Categories

```
SELECT TOP 3 *
FROM Categories
```

Table 3: 3 records

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads

Customers

```
SELECT TOP 3 *
FROM Customers
```

Table 4: 3 records

CustomerID	CompanyName	ContactName	ContactTitle	Address
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312

Employees

```
SELECT TOP 3 *
FROM Employees
```

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address
1	Davolio	Nancy	Sales Representative	Ms.	1966-12-08	2010-05-01	507 - 20th
Apt. 2A Sea	ttle WA	98122	USA (206) 55	5-9857 5467	Education	includes a BA	in psycho
2	Fuller	Andrew	Vice President, Sales	Dr.	1970-02-19	2010-08-14	908 W. C
3	Leverling	Janet	Sales Representative	Ms.	1981-08-30	2010-04-01	722 Moss

Order Details

```
SELECT TOP 3 *
FROM OrderDetails
```

Table 6: 3 records

OrderID	ProductID	UnitPrice	Quantity	Discount
10248	11	14.0	12	0
10248	42	9.8	10	0
10248	72	34.8	5	0

Orders

```
SELECT TOP 3 *
FROM Orders
```

Table 7: 3 records

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipNar
10248	VINET	5	2014-07-04 08:00:00	2014-08-01	2014-07-16	3	32.38	Vins et
10249	TOMSP	6	2014-07-05 04:00:00	2014-08-16	2014-07-10	1	11.61	Toms Sp
10250	HANAR	4	2014-07-08 15:00:00	2014-08-05	2014-07-12	2	65.83	Hanari

Products

```
SELECT TOP 3 *
FROM Products
```

Table 8: 3 records

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder
1	Chai	1	1	10 boxes x 20 bags	18	39	0
2	Chang	1	1	24 - 12 oz bottles	19	17	40

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10	13	70

Shippers

```
SELECT TOP 3 *
FROM Shippers
```

Table 9: 3 records

ShipperID	CompanyName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

Suppliers

```
SELECT TOP 3 *
FROM Suppliers
```

Table 10: 3 records

SupplierID	CompanyName	ContactName	ContactTitle	Address	City	R
1	Exotic Liquids	Charlotte Cooper	Purchasing Manager	49 Gilbert St.	London	N
2	New Orleans Cajun Delights	Shelley Burke	Order Administrator	P.O. Box 78934	New Orleans	L
3	Grandma Kelly's Homestead	Regina Murphy	Sales Representative	707 Oxford Rd.	Ann Arbor	M

Easy Queries

Functions Used:

- Like
  - Determins whether a specific character string matches a specified pattern.
- Wildcard %
  - Matches any single character within the specified range or set that is specified between brackets.
- In
  - Determines whether a specified value matches any value in a subquery or list
- Date
  - coerces to date object
- Order By
  - sorts data returned by a query
- Group By
  - A select statement clause that divides the query result into groups of rows, usually for the purpose of performing one or more aggregations on the group
- Convert
  - coerces to given type
- Concat

  - adds two strings together
- Count
  - returns the number of items in a group
- Min
  - returns the minimum value
- Distinct
  - returns unique values
- Join
  - joins two tables together on a specific column

1 | Which Shippers Do We Have?

Return all the fields from all the shippers.

```
SELECT *
FROM Shippers
```

Table 11: 3 records

ShipperID	CompanyName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

2 | Specific Fields from Categories

Get the categoryname and description from categories.

```
SELECT CategoryName, Description
FROM Categories
```

Table 12: 8 records

CategoryName	Description
Beverages	Soft drinks, coffees, teas, beers, and ales
Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
Confections	Desserts, candies, and sweet breads
Dairy Products	Cheeses
Grains/Cereals	Breads, crackers, pasta, and cereal
Meat/Poultry	Prepared meats
Produce	Dried fruit and bean curd
Seafood	Seaweed and fish

3 | Sales Representatives

Get first name, last name, hire date of all employees with the title of sales representative

```
SELECT FirstName, LastName, HireDate
FROM Employees
WHERE Title = 'Sales Representative'
```

Table 13: 6 records

FirstName	LastName	HireDate
Nancy	Davolio	2010-05-01
Janet	Leverling	2010-04-01
Margaret	Peacock	2011-05-03
Michael	Suyama	2011-10-17
Robert	King	2012-01-02
Anne	Dodsworth	2012-11-15

4 | Sales Representatives in the United States

Constrain to employees in the USA

```
SELECT FirstName, LastName, HireDate
FROM Employees
WHERE Title = 'Sales Representative'
AND Country = 'USA'
```

Table 14: 3 records

FirstName	LastName	HireDate
Nancy	Davolio	2010-05-01
Janet	Leverling	2010-04-01
Margaret	Peacock	2011-05-03

5 | Orders Placed by Specific EmployeeID

Show all orders places by a specific employee.

```
SELECT OrderID, OrderDate
FROM Orders
WHERE EmployeeID = 5
```

Table 15: Displaying records 1 - 25

OrderID	OrderDate
10248	2014-07-04 08:00:00
10254	2014-07-11 02:00:00
10269	2014-07-31 00:00:00
10297	2014-09-04 21:00:00
10320	2014-10-03 12:00:00
10333	2014-10-18 18:00:00
10358	2014-11-20 05:00:00
10359	2014-11-21 14:00:00
10372	2014-12-04 10:00:00
10378	2014-12-10 00:00:00
10397	2014-12-27 17:00:00
10463	2015-03-04 13:00:00
10474	2015-03-13 16:00:00
10477	2015-03-17 02:00:00

OrderID	OrderDate
10529	2015-05-07 01:00:00
10549	2015-05-27 03:00:00
10569	2015-06-16 15:00:00
10575	2015-06-20 22:00:00
10607	2015-07-22 09:00:00
10648	2015-08-28 22:00:00
10649	2015-08-28 00:00:00
10650	2015-08-29 06:00:00
10654	2015-09-02 07:00:00
10675	2015-09-19 06:00:00
10711	2015-10-21 03:00:00

6 | Suppliers and Contact Titles

In the suppliers table, we want the supplier ID, contact name and contact title for those suppliers whose contact title is not marketing manager

```
SELECT SupplierID, ContactName, ContactTitle
FROM Suppliers
WHERE NOT ContactTitle = 'Marketing Manager'
```

Table 16: 24 records

SupplierID	ContactName	ContactTitle
1	Charlotte Cooper	Purchasing Manager
2	Shelley Burke	Order Administrator
3	Regina Murphy	Sales Representative
5	Antonio del Valle Saavedra	Export Administrator
6	Mayumi Ohno	Marketing Representative
8	Peter Wilson	Sales Representative
9	Lars Peterson	Sales Agent
11	Petra Winkler	Sales Manager
12	Martin Bein	International Marketing Mgr.
13	Sven Petersen	Coordinator Foreign Markets
14	Elio Rossi	Sales Representative
16	Cheryl Saylor	Regional Account Rep.
17	Michael Björn	Sales Representative
18	Guylène Nodier	Sales Manager
19	Robb Merchant	Wholesale Account Agent
20	Chandra Leka	Owner
21	Niels Petersen	Sales Manager
22	Dirk Luchte	Accounting Manager
23	Anne Heikkonen	Product Manager
24	Wendy Mackenzie	Sales Representative
26	Giovanni Giudici	Order Administrator
27	Marie Delamare	Sales Manager
28	Eliane Noz	Sales Representative
29	Chantal Goulet	Accounting Manager

7 | Products with “Queso” in Product Name

We want the product ID and product name for those products where the product name includes the string Queso

```
SELECT ProductID, ProductName
FROM Products
WHERE ProductName
LIKE '%queso%'
```

Table 17: 2 records

ProductID	ProductName
11	Queso Cabrales
12	Queso Manchego La Pastora

8 | Orders Shipping to France or Belgium

We want the order id, customer id, and ship country for the orders where the ship country is either France or Belgium. The following table shows 25 / 96 rows.

```
SELECT OrderID, CustomerID, ShipCountry
FROM Orders
WHERE ShipCountry
IN ('France', 'Belgium')
```

Table 18: Displaying records 1 - 25

OrderID	CustomerID	ShipCountry
10248	VINET	France
10251	VICTE	France
10252	SUPRD	Belgium
10265	BLONP	France
10274	VINET	France
10295	VINET	France
10297	BLONP	France
10302	SUPRD	Belgium
10311	DUMON	France
10331	BONAP	France
10334	VICTE	France
10340	BONAP	France
10350	LAMAI	France
10358	LAMAI	France
10360	BLONP	France
10362	BONAP	France
10371	LAMAI	France
10408	FOLIG	France
10413	LAMAI	France
10425	LAMAI	France
10436	BLONP	France
10449	BLONP	France
10450	VICTE	France
10454	LAMAI	France

OrderID	CustomerID	ShipCountry
10458	SUPRD	Belgium

9 | Orders Shipping to any Country in Latin America

We want to get all the orders from any latin american country | We don't have a list of the Latin American countries in a table, so we will make it.

This table outputs 25 / 173 rows.

```
SELECT OrderID, CustomerID, ShipCountry
FROM Orders
WHERE ShipCountry
IN ('Brazil', 'Mexico', 'Argentina', 'Venezuela')
```

Table 19: Displaying records 1 - 25

OrderID	CustomerID	ShipCountry
10250	HANAR	Brazil
10253	HANAR	Brazil
10256	WELLI	Brazil
10257	HILAA	Venezuela
10259	CENTC	Mexico
10261	QUEDE	Brazil
10268	GROSR	Venezuela
10276	TORTU	Mexico
10283	LILAS	Venezuela
10287	RICAR	Brazil
10290	COMMI	Brazil
10291	QUEDE	Brazil
10292	TRADH	Brazil
10293	TORTU	Mexico
10296	LILAS	Venezuela
10299	RICAR	Brazil
10304	TORTU	Mexico
10308	ANATR	Mexico
10319	TORTU	Mexico
10322	PERIC	Mexico
10330	LILAS	Venezuela
10347	FAMIA	Brazil
10354	PERIC	Mexico
10357	LILAS	Venezuela
10365	ANTON	Mexico

10 | Employees in Order of Age

For all the employees, show first, last, title and birth date in order of birth date with oldest employees first

```
SELECT FirstName, LastName, Title, BirthDate
FROM Employees
ORDER BY BirthDate ASC
```

Table 20: 9 records

FirstName	LastName	Title	BirthDate
Margaret	Peacock	Sales Representative	1955-09-19
Nancy	Davolio	Sales Representative	1966-12-08
Andrew	Fuller	Vice President, Sales	1970-02-19
Steven	Buchanan	Sales Manager	1973-03-04
Laura	Callahan	Inside Sales Coordinator	1976-01-09
Robert	King	Sales Representative	1978-05-29
Michael	Suyama	Sales Representative	1981-07-02
Janet	Leverling	Sales Representative	1981-08-30
Anne	Dodsworth	Sales Representative	1984-01-27

11 | Showing Only the Date with a DateTime Field

In the above query, suppose we only want the date portion of the birth date field.

```
SELECT FirstName, LastName, Title, BDate = convert(DATE, BirthDate)
FROM Employees
ORDER BY BDate ASC
```

Table 21: 9 records

FirstName	LastName	Title	BDate
Margaret	Peacock	Sales Representative	1955-09-19
Nancy	Davolio	Sales Representative	1966-12-08
Andrew	Fuller	Vice President, Sales	1970-02-19
Steven	Buchanan	Sales Manager	1973-03-04
Laura	Callahan	Inside Sales Coordinator	1976-01-09
Robert	King	Sales Representative	1978-05-29
Michael	Suyama	Sales Representative	1981-07-02
Janet	Leverling	Sales Representative	1981-08-30
Anne	Dodsworth	Sales Representative	1984-01-27

12 | Employees Full Name

show first, last and firstlast names

```
SELECT FirstName, LastName, FullName = CONCAT(FirstName, ' ', LastName)
FROM Employees
```

Table 22: 9 records

FirstName	LastName	FullName
Nancy	Davolio	Nancy Davolio
Andrew	Fuller	Andrew Fuller
Janet	Leverling	Janet Leverling
Margaret	Peacock	Margaret Peacock
Steven	Buchanan	Steven Buchanan
Michael	Suyama	Michael Suyama
Robert	King	Robert King
Laura	Callahan	Laura Callahan

FirstName	LastName	FullName
Anne	Dodsworth	Anne Dodsworth

13 | Order Details Amount per Line Item

In the order details table we have the fields unit price and quantity. We want a new field, total price that multiplies these two together. We also want order id, product id, unit price, and quantity ordered by order id and product id.

This table outputs 25 / 2155 rows.

```
SELECT OrderID, ProductID, UnitPrice, Quantity, TotalPrice = UnitPrice * Quantity
FROM OrderDetails
ORDER BY OrderID, ProductID
```

Table 23: Displaying records 1 - 25

OrderID	ProductID	UnitPrice	Quantity	TotalPrice
10248	11	14.0	12	168.0
10248	42	9.8	10	98.0
10248	72	34.8	5	174.0
10249	14	18.6	9	167.4
10249	51	42.4	40	1696.0
10250	41	7.7	10	77.0
10250	51	42.4	35	1484.0
10250	65	16.8	15	252.0
10251	22	16.8	6	100.8
10251	57	15.6	15	234.0
10251	65	16.8	20	336.0
10252	20	64.8	40	2592.0
10252	33	2.0	25	50.0
10252	60	27.2	40	1088.0
10253	31	10.0	20	200.0
10253	39	14.4	42	604.8
10253	49	16.0	40	640.0
10254	24	3.6	15	54.0
10254	55	19.2	21	403.2
10254	74	8.0	21	168.0
10255	2	15.2	20	304.0
10255	16	13.9	35	486.5
10255	36	15.2	25	380.0
10255	59	44.0	30	1320.0
10256	53	26.2	15	393.0

14 | How Many Customers?

How many customers are there in the customers table?

```
SELECT TotalCustomers = COUNT(*)
FROM Customers
```

Table 24: 1 records

TotalCustomers
91

15 | Date of First Order

Return the date of the first order

```
SELECT FirstOrder = MIN(OrderDate)
FROM Orders
```

Table 25: 1 records

FirstOrder
2014-07-04 08:00:00

16 | Countries with Customers

We want a list of countries where Northwind has customers

```
SELECT DISTINCT(Country)
FROM Customers
```

Table 26: 21 records

Country
Argentina
Austria
Belgium
Brazil
Canada
Denmark
Finland
France
Germany
Ireland
Italy
Mexico
Norway
Poland
Portugal
Spain
Sweden
Switzerland
UK
USA
Venezuela

17 | Contact Titles for Customers

Return a list of all the different values in the customers table for contact titles. Also include a count for each contact title.

```
SELECT ContactTitle, TotalContactTitle = COUNT(*)
FROM Customers
GROUP BY ContactTitle
ORDER BY TotalContactTitle DESC
```

Table 27: 12 records

ContactTitle	TotalContactTitle
Owner	17
Sales Representative	17
Marketing Manager	12
Sales Manager	11
Accounting Manager	10
Sales Associate	7
Marketing Assistant	6
Sales Agent	5
Assistant Sales Agent	2
Order Administrator	2
Assistant Sales Representative	1
Owner/Marketing Assistant	1

18 | Products with Associated Supplier Names

For each product, we want the associated supplier. Show the product id, product name, and company name of the supplier and sort by product id.

This table outputs 25 / 77 rows.

```
SELECT ProductID, ProductName, Supplier = CompanyName
FROM Products
JOIN Suppliers ON Products.SupplierID = Suppliers.SupplierID
```

Table 28: Displaying records 1 - 25

ProductID	ProductName	Supplier
1	Chai	Exotic Liquids
2	Chang	Exotic Liquids
3	Aniseed Syrup	Exotic Liquids
4	Chef Anton's Cajun Seasoning	New Orleans Cajun Delights
5	Chef Anton's Gumbo Mix	New Orleans Cajun Delights
6	Grandma's Boysenberry Spread	Grandma Kelly's Homestead
7	Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead
8	Northwoods Cranberry Sauce	Grandma Kelly's Homestead
9	Mishi Kobe Niku	Tokyo Traders
10	Ikura	Tokyo Traders
11	Queso Cabrales	Cooperativa de Quesos 'Las Cabras'
12	Queso Manchego La Pastora	Cooperativa de Quesos 'Las Cabras'
13	Konbu	Mayumi's
14	Tofu	Mayumi's

ProductID	ProductName	Supplier
15	Genen Shouyu	Mayumi's
16	Pavlova	Pavlova, Ltd.
17	Alice Mutton	Pavlova, Ltd.
18	Carnarvon Tigers	Pavlova, Ltd.
19	Teatime Chocolate Biscuits	Specialty Biscuits, Ltd.
20	Sir Rodney's Marmalade	Specialty Biscuits, Ltd.
21	Sir Rodney's Scones	Specialty Biscuits, Ltd.
22	Gustaf's Knäckebröd	PB Knäckebröd AB
23	Tunnbröd	PB Knäckebröd AB
24	Guaraná Fantástica	Refrescos Americanas LTDA
25	NuNuCa Nuß-Nougat-Creme	Heli Süßwaren GmbH & Co. KG

19 | Orders and the Shipper Used

We'd like to show a list of the orders that were made, including the shipper that was used. We want order id, order date (date only), company name of the shipper, and to sort by order id with order id < 10270.

```
SELECT OrderID, OrderDate = CONVERT(DATE, ORDERDATE), Shipper = CompanyName
FROM Orders JOIN Shippers ON Orders.ShipVia = Shippers.ShipperID
WHERE OrderID < 10270 ORDER BY OrderID
```

Table 29: 22 records

OrderID	OrderDate	Shipper
10248	2014-07-04	Federal Shipping
10249	2014-07-05	Speedy Express
10250	2014-07-08	United Package
10251	2014-07-08	Speedy Express
10252	2014-07-09	United Package
10253	2014-07-10	United Package
10254	2014-07-11	United Package
10255	2014-07-12	Federal Shipping
10256	2014-07-15	United Package
10257	2014-07-16	Federal Shipping
10258	2014-07-17	Speedy Express
10259	2014-07-18	Federal Shipping
10260	2014-07-19	Speedy Express
10261	2014-07-19	United Package
10262	2014-07-22	Federal Shipping
10263	2014-07-23	Federal Shipping
10264	2014-07-24	Federal Shipping
10265	2014-07-25	Speedy Express
10266	2014-07-26	Federal Shipping
10267	2014-07-29	Speedy Express
10268	2014-07-30	Federal Shipping
10269	2014-07-31	Speedy Express

Intermediate Problems

Functions Used:

- Exists
  - specifies a subquery to test for the existence of rows
- DateAdd
  - Adds a specified number value to a specified datepart of an input date value, and then returns that modified value
- Nested Select
  - allows for a query as an operated on dataframe
- Multiple Join
  - joins multiple tables on specified column values
- Date
  - coerces to date
- Year
  - pulls the year value from a date object
- Top
  - returns the top n rows
- Case
  - allows for a control sequence on data
- Count
  - returns number of items from a group
- Order By
  - sorts the data
- Group By
  - A select statement clause that divides the query result into groups of rows, usually for the purpose of performing one or more aggregations on the group
- Convert
  - coerces to a given datatype
- Avg
  - returns the average of a column of numbers
- Not
  - returns the negation of the statement

20 | Categories and Total Products in Each Category

We want to see the total number of products in each category and sort the results by the total number of products in descending order

```
SELECT CategoryName, TotalProducts = COUNT(*)
FROM Products JOIN Categories ON Products.CategoryID = Categories.CategoryID
GROUP BY CategoryName
ORDER BY TotalProducts DESC
```

Table 30: 8 records

CategoryName	TotalProducts
Confections	13
Beverages	12
Condiments	12
Seafood	12
Dairy Products	10
Grains/Cereals	7

CategoryName	TotalProducts
Meat/Poultry	6
Produce	5

21 | Total Customers per Country / City

Show the total number of customers per country and city.

This table output 25 / 69 rows.

```
SELECT Country, City, TotalCustomers = Count(*)
FROM Customers
GROUP BY Country, City
ORDER BY TotalCustomers DESC
```

Table 31: Displaying records 1 - 25

Country	City	TotalCustomers
UK	London	6
Mexico	México D.F.	5
Brazil	Sao Paulo	4
Brazil	Rio de Janeiro	3
Spain	Madrid	3
Argentina	Buenos Aires	3
France	Paris	2
USA	Portland	2
France	Nantes	2
Portugal	Lisboa	2
Finland	Oulu	1
Italy	Reggio Emilia	1
France	Reims	1
Brazil	Resende	1
Austria	Salzburg	1
Venezuela	San Cristóbal	1
USA	San Francisco	1
USA	Seattle	1
Spain	Sevilla	1
Norway	Stavern	1
France	Strasbourg	1
Germany	Stuttgart	1
Italy	Torino	1
France	Toulouse	1
Canada	Tsawassen	1

22 | Products that Need Reordering

We want to find the products in our inventory that should be reordered. We can use the fields UnitsInStock and ReorderLEvel where UnitsInStock < ReorderLevel. Sort the results by Product ID

```
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
WHERE UnitsInStock <= ReorderLevel
```

```
ORDER BY ProductID
```

Table 32: 22 records

ProductID	ProductName	UnitsInStock	ReorderLevel
2	Chang	17	25
3	Aniseed Syrup	13	25
5	Chef Anton's Gumbo Mix	0	0
11	Queso Cabrales	22	30
17	Alice Mutton	0	0
21	Sir Rodney's Scones	3	5
29	Thüringer Rostbratwurst	0	0
30	Nord-Ost Matjeshering	10	15
31	Gorgonzola Telino	0	20
32	Mascarpone Fabioli	9	25
37	Gravad lax	11	25
43	Ipoh Coffee	17	25
45	Rogede sild	5	15
48	Chocolade	15	25
49	Maxilaku	10	15
53	Perth Pasties	0	0
56	Gnocchi di nonna Alice	21	30
64	Wimmers gute Semmelknödel	22	30
66	Louisiana Hot Spiced Okra	4	20
68	Scottish Longbreads	6	15
70	Outback Lager	15	30
74	Longlife Tofu	4	5

23 | Products that Need Reordering, Continued

Now we need to incorporate the fields UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued into our calculation.

We can define products that need reordering with the following:

- UnitsInStock + UnitsOnOrder are less than or equal to ReorderLevel
- The discontinued flag is false (0)

```
SELECT ProductID, ProductName, UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued
FROM Products
WHERE NOT (UnitsInStock + UnitsOnOrder) > ReorderLevel
AND Discontinued = 0
```

Table 33: 2 records

ProductID	ProductName	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
30	Nord-Ost Matjeshering	10	0	15	FALSE
70	Outback Lager	15	10	30	FALSE

24 | Customer List by Region

A salesperson from Northwind is going on a business trip to visit customers, and would like to see a list of all customers, sorted by region, alphabetically.

However, he wants the customers with no region (null region) to be at the end instead of at the top. Within the same region, companies should be sorted by customer id.

This table outputs 25 / 96 rows.

```
SELECT CustomerID, CompanyName, Region
FROM Customers
ORDER BY
CASE
  WHEN Region IS NULL THEN 1
  ELSE 0
END, Region
ASC
```

Table 34: Displaying records 1 - 25

CustomerID	CompanyName	Region
OLDWO	Old World Delicatessen	AK
LAUGB	Laughing Bacchus Wine Cellars	BC
BOTTM	Bottom-Dollar Markets	BC
LETSS	Let's Stop N Shop	CA
HUNGO	Hungry Owl All-Night Grocers	Co. Cork
GROSR	GROSELLA-Restaurante	DF
SAVEA	Save-a-lot Markets	ID
ISLAT	Island Trading	Isle of Wight
LILAS	LILA-Supermercado	Lara
THECR	The Cracker Box	MT
RATTC	Rattlesnake Canyon Grocery	NM
LINOD	LINO-Delicateses	Nueva Esparta
LONEP	Lonesome Pine Restaurant	OR
HUNGC	Hungry Coyote Import Store	OR
THEBI	The Big Cheese	OR
GREAL	Great Lakes Food Market	OR
MEREP	Mère Paillarde	Québec
QUEDE	Que Delícia	RJ
RICAR	Ricardo Adocicados	RJ
HANAR	Hanari Carnes	RJ
GOURL	Gourmet Lanchonetes	SP
FAMIA	Familia Arquibaldo	SP
COMMI	Comércio Mineiro	SP
QUEEN	Queen Cozinha	SP
TRADH	Tradição Hipermercados	SP

25 | High Freight Charges

Some of the countries shipped to have very high freight charges. Return the three ship countries with the highest average freight overall, in descending order by average freight.

```sql
SELECT TOP 3 ShipCountry, AverageFreight = AVG(Freight)
FROM Orders
GROUP BY ShipCountry
ORDER BY AverageFreight DESC
```

Table 35: 3 records

ShipCountry	AverageFreight
Austria	184.7875
Ireland	145.0126
USA	112.8794

26 | High Freight Charges for 2015

Lets return the highest average freight charges from 2015

```sql
SELECT TOP 3 ShipCountry, AverageFreight = AVG(Freight)
FROM Orders
WHERE YEAR(CONVERT(DATE, OrderDate)) = 2015
GROUP BY ShipCountry
ORDER BY AverageFreight DESC
```

Table 36: 3 records

ShipCountry	AverageFreight
Austria	178.3642
Switzerland	117.1775
France	113.9910

27 | High Freight Charges Last Year

We want to see the top three ship countries with highest average freight charges again, but this time we want to use the last 12 months of order data.

```sql
SELECT TOP 3 ShipCountry, AverageFreight = AVG(Freight)
FROM Orders
WHERE OrderDate BETWEEN DATEADD(yy, -1, (SELECT MAX(OrderDate) FROM Orders))
AND (SELECT MAX(OrderDate) FROM Orders)
GROUP BY ShipCountry
ORDER BY AverageFreight DESC
```

Table 37: 3 records

ShipCountry	AverageFreight
Ireland	200.2100
Austria	186.4596
USA	119.3032

28 | Employee / Order Detail Report

For inventory, we need to show employee and order detail information for all orders. Sort by Order ID and Product ID.

The table below shows 25 / 2155 rows.

```
SELECT Employees.EmployeeID, LastName, Orders.OrderID, Products.ProductName, OrderDetails.Quantity
FROM Employees
JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
ORDER BY Orders.OrderID, Products.ProductID
```

Table 38: Displaying records 1 - 25

EmployeeID	LastName	OrderID	ProductName	Quantity
5	Buchanan	10248	Queso Cabrales	12
5	Buchanan	10248	Singaporean Hokkien Fried Mee	10
5	Buchanan	10248	Mozzarella di Giovanni	5
6	Suyama	10249	Tofu	9
6	Suyama	10249	Manjimup Dried Apples	40
4	Peacock	10250	Jack's New England Clam Chowder	10
4	Peacock	10250	Manjimup Dried Apples	35
4	Peacock	10250	Louisiana Fiery Hot Pepper Sauce	15
3	Leverling	10251	Gustaf's Knäckebröd	6
3	Leverling	10251	Ravioli Angelo	15
3	Leverling	10251	Louisiana Fiery Hot Pepper Sauce	20
4	Peacock	10252	Sir Rodney's Marmalade	40
4	Peacock	10252	Geitost	25
4	Peacock	10252	Camembert Pierrot	40
3	Leverling	10253	Gorgonzola Telino	20
3	Leverling	10253	Chartreuse verte	42
3	Leverling	10253	Maxilaku	40
5	Buchanan	10254	Guaraná Fantástica	15
5	Buchanan	10254	Pâté chinois	21
5	Buchanan	10254	Longlife Tofu	21
9	Dodsworth	10255	Chang	20
9	Dodsworth	10255	Pavlova	35
9	Dodsworth	10255	Inlagd Sill	25
9	Dodsworth	10255	Raclette Courdavault	30
3	Leverling	10256	Perth Pasties	15

29 | Customers with No Orders

We want to return customers who have never placed an order.

```
SELECT Customers_Customer_ID = Customers.CustomerID, Orders_Customer_ID = Orders.CustomerID
FROM Customers
LEFT JOIN Orders ON Orders.CustomerID = Customers.CustomerID
WHERE Orders.CustomerID is NULL
```

Table 39: 2 records

Customers_Customer_ID	Orders_Customer_ID
PARIS	NA
FISSA	NA

30 | Customers with No Orders for EmployeeID 4

One employee has placed the most orders (EmployeeID 4: Margaret Peacock). However, there are still some customers who have not placed an order with her. We want to show those customers.

```
SELECT CustomerID
FROM Customers
WHERE NOT EXISTS
  (SELECT CustomerID
   FROM Orders
   WHERE Orders.CustomerID = Customers.CustomerID
   AND EmployeeID = 4)
```

Table 40: 16 records

CustomerID
CONSH
DUMON
FISSA
FRANR
GROSR
LAUGB
LAZYK
NORTS
PARIS
PERIC
PRINI
SANTG
SEVES
SPECD
THEBI
VINET

Advanced Problems

Functions Used:

- sum
  - returns a sum of a column of numeric values
- multiple join
  - joins multiple tables by column
- convert
  - coerces to a given datatype
- date
  - coerces to a date object

- group by
  - A select statement clause that divides the query result into groups of rows, usually for the purpose of performing one or more aggregations on the group
- order by
  - sorts the given data by specified values
- round
  - rounds to nearest given integer expansion, e.g. whole number, 0.1, 0.01, 0.001, ...
- eomonth
  - returns the last day of each months value
- top
  - returns the top n rows
- newid
  - creates a unique identifier for the assigned value
- percent
  - returns a percentage of the data as opposed to a fixed integer
- count
  - returns a sum of a specified data set
- subquery
  - a query within a query
- common table expressions
  - a query created specifically for aid in another query
- distinct
  - returns unique values
- isnull
  - if a value is null, coerces null to a given number
- case
  - allows for control flow on a given computation
- union
  - a set operation that returns all unique values of multiple given queries
- full outer join
  - a join which keeps the values of both joined tables, regardless of whether the items are in both tables or not
- row_number
  - adds a row number identifier
- window partition
  - partitions the data into segments and allows for a set of functions to be performed over the partitions
- table alias
  - allows for multiple named instances of a single database to be modified independently and used in a query

31 | High Value Customers

We want to find the customers who have made at least 1 order with a total value (not including discount) of $10,000 or more. We are only considering purchases made in 2016

```
SELECT Customers.CustomerID, Customers.CompanyName, Orders.OrderID, TotalPrice = SUM(OrderDetails.UnitP
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
WHERE YEAR(CONVERT(DATE, OrderDate)) = 2016
GROUP BY Customers.CustomerID, Customers.CompanyName, Orders.OrderID
HAVING SUM(OrderDetails.UnitPrice * OrderDetails.Quantity) > 10000
```

```
ORDER BY TotalPrice DESC
```

Table 41: 6 records

CustomerID	CompanyName	OrderID	TotalPrice
QUICK	QUICK-Stop	10865	17250.00
SAVEA	Save-a-lot Markets	11030	16321.90
HANAR	Hanari Carnes	10981	15810.00
KOENE	Königlich Essen	10817	11490.70
RATTC	Rattlesnake Canyon Grocery	10889	11380.00
HUNGO	Hungry Owl All-Night Grocers	10897	10835.24

32 | High Value Customers - Total Orders

We want to redefine high value customers as those who have orders totalling $15,000 or more in 2016.

```
SELECT Customers.CustomerID, Customers.CompanyName, TotalPrice = SUM(OrderDetails.UnitPrice * OrderDeta
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
WHERE YEAR(CONVERT(DATE, OrderDate)) = 2016
GROUP BY Customers.CustomerID, Customers.CompanyName
HAVING SUM(OrderDetails.UnitPrice * OrderDetails.Quantity) > 15000
ORDER BY TotalPrice DESC
```

Table 42: 9 records

CustomerID	CompanyName	TotalPrice
SAVEA	Save-a-lot Markets	42806.25
ERNSH	Ernst Handel	42598.90
QUICK	QUICK-Stop	40526.99
HANAR	Hanari Carnes	24238.05
HUNGO	Hungry Owl All-Night Grocers	22796.34
RATTC	Rattlesnake Canyon Grocery	21725.60
KOENE	Königlich Essen	20204.95
FOLKO	Folk och fä HB	15973.85
WHITC	White Clover Markets	15278.90

33 | High Value Customers with Discount

We wish to calcuate high value customers and order by the total amount including discount.

This table shows 25/81 rows.

```
SELECT Customers.CustomerID, Customers.CompanyName,
       TotalsWithoutDiscount = SUM(OrderDetails.UnitPrice * OrderDetails.Quantity),
       TotalsWithDiscount = ROUND(SUM(OrderDetails.UnitPrice * OrderDetails.Quantity * (1 - OrderDetails
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
WHERE YEAR(CONVERT(DATE, OrderDate)) = 2016
GROUP BY Customers.CustomerID, Customers.CompanyName
```

```
ORDER BY TotalsWithDiscount DESC
```

Table 43: Displaying records 1 - 25

CustomerID	CompanyName	TotalsWithoutDiscount	TotalsWithDiscount
ERNSH	Ernst Handel	42598.90	41210.65
QUICK	QUICK-Stop	40526.99	37217.32
SAVEA	Save-a-lot Markets	42806.25	36310.11
HANAR	Hanari Carnes	24238.05	23821.20
RATTC	Rattlesnake Canyon Grocery	21725.60	21238.27
HUNGO	Hungry Owl All-Night Grocers	22796.34	20402.12
KOENE	Königlich Essen	20204.95	19582.77
WHITC	White Clover Markets	15278.90	15278.90
FOLKO	Folk och fä HB	15973.85	13644.07
SUPRD	Suprêmes délices	11862.50	11644.60
BOTTM	Bottom-Dollar Markets	12227.40	11338.55
GREAL	Great Lakes Food Market	10562.58	9942.14
EASTC	Eastern Connection	9569.31	9296.68
LINOD	LINO-Delicateses	10085.60	9117.09
RICAR	Ricardo Adocicados	7312.00	6998.53
GODOS	Godos Cocina Típica	7064.05	6870.21
BERGS	Berglunds snabbköp	8110.55	6754.16
BONAP	Bon app'	7185.90	6680.61
QUEEN	Queen Cozinha	7007.65	6373.83
HILAA	HILARION-Abastos	6132.30	6043.20
AROUT	Around the Horn	5838.50	5604.75
LILAS	LILA-Supermercado	5994.06	5507.32
FRANK	Frankenversand	5587.00	5078.74
OLDWO	Old World Delicatessen	5337.65	5026.29
RICSU	Richter Supermarkt	5497.90	4988.85

34 | Month End Orders

At the end of the month, sales people may try to get more orders to meet a quota. Show all the orders made on the last day of the month, and order it by the employee ID and order ID.

```
SELECT EmployeeID, OrderID, OrderDate
FROM Orders
WHERE OrderDate = EOMONTH(OrderDate)
ORDER BY EmployeeID, OrderID
```

Table 44: 24 records

EmployeeID	OrderID	OrderDate
1	10461	2015-02-28
1	10616	2015-07-31
2	10583	2015-06-30
2	10686	2015-09-30
2	10989	2016-03-31
2	11060	2016-04-30
3	10432	2015-01-31
3	10988	2016-03-31

EmployeeID	OrderID	OrderDate
3	11063	2016-04-30
4	10343	2014-10-31
4	10522	2015-04-30
4	10584	2015-06-30
4	10617	2015-07-31
4	10725	2015-10-31
4	11061	2016-04-30
4	11062	2016-04-30
5	10269	2014-07-31
6	10317	2014-09-30
7	10490	2015-03-31
8	10399	2014-12-31
8	10460	2015-02-28
8	10491	2015-03-31
8	10987	2016-03-31
9	10687	2015-09-30

35 | Orders with Many Line Items

The mobile app developers are testing an app that customers will use to show orders. In order to make sure that even the largest orders show up correctly on the app, they'd like some samples of orders that have lots of individual line items. Show the top 10 orders with the most line items, in order of total line items.

```
SELECT TOP 10 Orders.OrderID, TotalOrderDetails = COUNT(*)
FROM Orders
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
GROUP BY Orders.OrderID
ORDER BY COUNT(*) DESC
```

Table 45: 10 records

OrderID	TotalOrderDetails
11077	25
10657	6
10847	6
10979	6
10273	5
10294	5
10309	5
10324	5
10325	5
10337	5

36 | Orders - Random Assortment

The app developers would like to get a random assortment of orders for beta testing on their app. Show a random set of 2% of all orders.

```
SELECT TOP 2 PERCENT OrderID
FROM Orders
```

```
ORDER BY NEWID()
```

Table 46: 17 records

OrderID
10345
10396
10998
10860
10509
10927
10902
10779
10473
10464
10318
10366
10664
10689
11048
10665
11019

37 | Orders - Accidental Double Entry

Suppose someone came to you claiming they accidentally entered a line item twice on an order, each time with a different product ID, but the same quantity. They remember the quantity was 60 or more. Show all the order IDs that match this, in order of order ID.

```
SELECT OrderID, Quantity
FROM OrderDetails
WHERE Quantity >= 60
GROUP BY OrderID, Quantity
HAVING COUNT(OrderID) > 1
ORDER BY OrderID
```

Table 47: 5 records

OrderID	Quantity
10263	60
10263	65
10658	70
10990	65
11030	100

38 | Orders - Accidental Double Entry with Details

We want to show the details of the order for orders that match the criteria of the above query.

```
;WITH Acc_Doub_Entry AS(
  SELECT OrderID, Quantity
```

```
  FROM OrderDetails
  WHERE Quantity >= 60
  GROUP BY OrderID, Quantity
  HAVING COUNT(OrderID) > 1)
SELECT OrderID, ProductID, UnitPrice, Quantity, Discount
FROM OrderDetails
WHERE OrderID IN (SELECT OrderID FROM Acc_Doub_Entry)
ORDER BY OrderID, Quantity
```

Table 48: 16 records

OrderID	ProductID	UnitPrice	Quantity	Discount
10263	16	13.90	60	0.25
10263	30	20.70	60	0.25
10263	24	3.60	65	0.00
10263	74	8.00	65	0.25
10658	60	34.00	55	0.05
10658	21	10.00	60	0.00
10658	40	18.40	70	0.05
10658	77	13.00	70	0.05
10990	34	14.00	60	0.15
10990	21	10.00	65	0.00
10990	55	24.00	65	0.15
10990	61	28.50	66	0.15
11030	29	123.79	60	0.25
11030	5	21.35	70	0.00
11030	2	19.00	100	0.25
11030	59	55.00	100	0.25

39 | Orders - Accidental Double Entry Details Derived Table

How would we get the same results as above without a common table expression?

```
SELECT OrderDetails.OrderID, ProductID, UnitPrice, Quantity, Discount
FROM OrderDetails
JOIN (SELECT DISTINCT(OrderID)
FROM OrderDetails
WHERE Quantity >= 60
GROUP BY OrderID, Quantity
HAVING COUNT(OrderID) > 1) PotentialProblemOrders
ON PotentialProblemOrders.OrderID = OrderDetails.OrderID
ORDER BY OrderID, ProductID
```

Table 49: 16 records

OrderID	ProductID	UnitPrice	Quantity	Discount
10263	16	13.90	60	0.25
10263	24	3.60	65	0.00
10263	30	20.70	60	0.25
10263	74	8.00	65	0.25
10658	21	10.00	60	0.00
10658	40	18.40	70	0.05

OrderID	ProductID	UnitPrice	Quantity	Discount
10658	60	34.00	55	0.05
10658	77	13.00	70	0.05
10990	21	10.00	65	0.00
10990	34	14.00	60	0.15
10990	55	24.00	65	0.15
10990	61	28.50	66	0.15
11030	2	19.00	100	0.25
11030	5	21.35	70	0.00
11030	29	123.79	60	0.25
11030	59	55.00	100	0.25

40 | Late Orders

Find all the orders that are arriving late. Place them in order by the longest delays to the shortest delays.

The table shows 25/39 rows.

```
SELECT OrderID, OrderDate = CONVERT(DATE, OrderDate), RequiredDate, ShippedDate
FROM Orders
WHERE CONVERT(DATE, ShippedDate) >= CONVERT(DATE, RequiredDate)
ORDER BY DATEDIFF(dd, CONVERT(DATE, ShippedDate), CONVERT(DATE, RequiredDate)) ASC
```

Table 50: Displaying records 1 - 25

OrderID	OrderDate	RequiredDate	ShippedDate
10777	2015-12-15	2015-12-29	2016-01-21
10726	2015-11-03	2015-11-17	2015-12-05
10423	2015-01-23	2015-02-06	2015-02-24
10970	2016-03-24	2016-04-07	2016-04-24
10515	2015-04-23	2015-05-07	2015-05-23
10827	2016-01-12	2016-01-26	2016-02-06
10660	2015-09-08	2015-10-06	2015-10-15
10663	2015-09-10	2015-09-24	2015-10-03
10828	2016-01-13	2016-01-27	2016-02-04
10924	2016-03-04	2016-04-01	2016-04-08
10451	2015-02-19	2015-03-05	2015-03-12
10545	2015-05-22	2015-06-19	2015-06-26
10593	2015-07-09	2015-08-06	2015-08-13
10427	2015-01-27	2015-02-24	2015-03-03
10380	2014-12-12	2015-01-09	2015-01-16
10309	2014-09-19	2014-10-17	2014-10-23
10927	2016-03-05	2016-04-02	2016-04-08
10960	2016-03-19	2016-04-02	2016-04-08
10705	2015-10-15	2015-11-12	2015-11-18
10709	2015-10-17	2015-11-14	2015-11-20
10847	2016-01-22	2016-02-05	2016-02-10
10727	2015-11-03	2015-12-01	2015-12-05
10596	2015-07-11	2015-08-08	2015-08-12
10483	2015-03-24	2015-04-21	2015-04-25
10578	2015-06-24	2015-07-22	2015-07-25

41 | Late Orders by Employees

Some salespeople have more orders arriving late than others. Which sales people have the most orders arriving late?

```
SELECT Employees.EmployeeID, Employees.LastName, TotalLateOrders = COUNT(*)
FROM Orders
JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
WHERE CONVERT(DATE, ShippedDate) >= CONVERT(DATE, RequiredDate)
GROUP BY Employees.EmployeeID, Employees.LastName
ORDER BY TotalLateOrders DESC
```

Table 51: 8 records

EmployeeID	LastName	TotalLateOrders
4	Peacock	10
3	Leverling	5
8	Callahan	5
9	Dodsworth	5
7	King	4
2	Fuller	4
1	Davolio	3
6	Suyama	3

42 | Late Orders vs. Total Orders

We want to compare the number of late orders per salesperson with the number of total orders per salesperson.

```
;WITH Tot_Late_Orders AS(
  SELECT Employees.EmployeeID, Employees.LastName, TotalLateOrders = COUNT(*)
  FROM Orders
  JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
  WHERE CONVERT(DATE, ShippedDate) >= CONVERT(DATE, RequiredDate)
  GROUP BY Employees.EmployeeID, Employees.LastName
  ),

Overall_Orders AS(
  SELECT Orders.EmployeeID, Employees.LastName, TotalOrders = COUNT(*)
  FROM Orders
  JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
  GROUP BY Orders.EmployeeID, Employees.LastName
)

SELECT Overall_Orders.EmployeeID, Overall_Orders.LastName, AllOrders = TotalOrders,
       LateOrders = ISNULL(TotalLateOrders, 0),
       Prop = ROUND(((ISNULL(TotalLateOrders, 0) * 1.0) / Overall_Orders.TotalOrders), 2)
FROM Tot_Late_Orders
RIGHT JOIN Overall_Orders ON Tot_Late_Orders.EmployeeID = Overall_Orders.EmployeeID
ORDER BY Prop DESC
```

EmployeeID	LastName	AllOrders	LateOrders	Prop

Table 52: 9 records

EmployeeID	LastName	AllOrders	LateOrders	Prop
9	Dodsworth	43	5	0.12
7	King	72	4	0.06
4	Peacock	156	10	0.06
8	Callahan	104	5	0.05
2	Fuller	96	4	0.04
3	Leverling	127	5	0.04
6	Suyama	67	3	0.04
1	Davolio	123	3	0.02
5	Buchanan	42	0	0.00

43 | Customer Grouping

The VP of Sales would like us to create customer groups depending on how much they ordered in 2016. The groupings would look like the following:

- 0 - 1000
- 1001 - 5000
- 5001 - 10000
- 10000+

The table below shows 25/81 rows.

```
SELECT Customers.CustomerID, Customers.CompanyName,
       TotalPrice = SUM(OrderDetails.UnitPrice * OrderDetails.Quantity),
       CustomerGroup = CASE
       WHEN SUM(OrderDetails.UnitPrice * OrderDetails.Quantity) <= 1000 THEN 'Low'
       WHEN SUM(OrderDetails.UnitPrice * OrderDetails.Quantity) > 1000
       AND SUM(OrderDetails.UnitPrice * OrderDetails.Quantity) <= 5000 THEN 'Medium'
       WHEN SUM(OrderDetails.UnitPrice * OrderDetails.Quantity) > 5000
       AND SUM(OrderDetails.UnitPrice * OrderDetails.Quantity) <= 10000 THEN 'High'
       ELSE 'Very High'
END
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
WHERE YEAR(CONVERT(DATE, OrderDate)) = 2016
GROUP BY Customers.CustomerID, Customers.CompanyName
ORDER BY Customers.CustomerID ASC
```

Table 53: Displaying records 1 - 10

CustomerID	CompanyName	TotalPrice	CustomerGroup
ALFKI	Alfreds Futterkiste	2302.20	Medium
ANATR	Ana Trujillo Emparedados y helados	514.40	Low
ANTON	Antonio Moreno Taquería	660.00	Low
AROUT	Around the Horn	5838.50	High
BERGS	Berglunds snabbköp	8110.55	High
BLAUS	Blauer See Delikatessen	2160.00	Medium

CustomerID	CompanyName	TotalPrice	CustomerGroup
BLONP	Blondesddsl père et fils	730.00	Low
BOLID	Bólido Comidas preparadas	280.00	Low
BONAP	Bon app'	7185.90	High
BOTTM	Bottom-Dollar Markets	12227.40	Very High

44 | Customer Grouping with Percentage

We would like to show the breakdown of each group and their respective percentages of the whole catalog of business.

```
;WITH Orders_2016 AS(
  SELECT Customers.CustomerID, Customers.CompanyName, TotalOrderAmount = SUM(Quantity * UnitPrice)
  FROM Customers
  JOIN Orders ON Orders.CustomerID = Customers.CustomerID
  JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
  WHERE YEAR(CONVERT(DATE, OrderDate)) = 2016
  GROUP BY Customers.CustomerID, Customers.CompanyName
),
Cust_Groups AS(
  SELECT CustomerID, CompanyName, TotalOrderAmount, CustomerGroup = CASE
    WHEN TotalOrderAmount <= 1000 THEN 'Low'
    WHEN TotalOrderAmount > 1000 AND TotalOrderAmount <= 5000 THEN 'Medium'
    WHEN TotalOrderAmount > 5000 AND TotalOrderAmount <= 10000 THEN 'High'
    ELSE 'Very High'
  END
  FROM Orders_2016)

SELECT CustomerGroup, TotalInGroup = COUNT(*),
       PercentageInGroup = COUNT(*) * 1.0 / (SELECT COUNT(*) FROM Cust_Groups)
FROM Cust_Groups
GROUP BY CustomerGroup
ORDER BY TotalInGroup DESC
```

Table 54: 4 records

CustomerGroup	TotalInGroup	PercentageInGroup
Medium	35	0.4320988
Low	20	0.2469136
High	13	0.1604938
Very High	13	0.1604938

45 | Customer Grouping - Flexible

The sales team would like an overview of the customer groupings, and would like them in a way in which they don't need to edit much sql in order to change the boundaries.

In this table we will use the CustomerGroupThresholds table. The table output is 25/81 rows.

```
;WITH Orders_2016 AS (
  SELECT Customers.CustomerID, Customers.CompanyName, TotalOrderAmount = SUM(Quantity * UnitPrice)
  FROM Customers
```

```
  JOIN Orders ON Orders.CustomerID = Customers.CustomerID
  JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
  WHERE YEAR(CONVERT(DATE, OrderDate)) = 2016
  GROUP BY Customers.CustomerID, Customers.CompanyName
)
SELECT CustomerID, CompanyName, TotalOrderAmount, CustomerGroupName
FROM Orders_2016
JOIN CustomerGroupThresholds ON Orders_2016.TotalOrderAmount
BETWEEN CustomerGroupThresholds.RangeBottom AND CustomerGroupThresholds.RangeTop
ORDER BY CustomerID
```

Table 55: Displaying records 1 - 25

CustomerID	CompanyName	TotalOrderAmount	CustomerGroupName
ALFKI	Alfreds Futterkiste	2302.20	Medium
ANATR	Ana Trujillo Emparedados y helados	514.40	Low
ANTON	Antonio Moreno Taquería	660.00	Low
AROUT	Around the Horn	5838.50	High
BERGS	Berglunds snabbköp	8110.55	High
BLAUS	Blauer See Delikatessen	2160.00	Medium
BLONP	Blondesddsl père et fils	730.00	Low
BOLID	Bólido Comidas preparadas	280.00	Low
BONAP	Bon app'	7185.90	High
BOTTM	Bottom-Dollar Markets	12227.40	Very High
BSBEV	B's Beverages	2431.00	Medium
CACTU	Cactus Comidas para llevar	1576.80	Medium
CHOPS	Chop-suey Chinese	4429.40	Medium
COMMI	Comércio Mineiro	513.75	Low
CONSH	Consolidated Holdings	931.50	Low
DRACD	Drachenblut Delikatessen	2809.61	Medium
DUMON	Du monde entier	860.10	Low
EASTC	Eastern Connection	9569.31	High
ERNSH	Ernst Handel	42598.90	Very High
FOLKO	Folk och fä HB	15973.85	Very High
FRANK	Frankenversand	5587.00	High
FRANR	France restauration	2252.06	Medium
FRANS	Franchi S.p.A.	1296.00	Medium
FURIB	Furia Bacalhau e Frutos do Mar	68.00	Low
GALED	Galería del gastrónomo	207.50	Low

46 | Countries with Suppliers or Customers

Return a list of all countries where suppliers or customers are based.

```
SELECT Distinct(Country)
FROM Customers
UNION
SELECT DISTINCT(Country)
FROM Suppliers
ORDER BY Country
```

Table 56: Displaying records 1 - 25

Country
Argentina
Australia
Austria
Belgium
Brazil
Canada
Denmark
Finland
France
Germany
Ireland
Italy
Japan
Mexico
Netherlands
Norway
Poland
Portugal
Singapore
Spain
Sweden
Switzerland
UK
USA
Venezuela

47 | Countries with Suppliers or Customers V2

With More Details

```
;WITH SupplierCountries AS(
SELECT DISTINCT(Country) FROM Suppliers
),
CustomerCountries AS(
SELECT DISTINCT(Country) FROM Customers
)
SELECT SupplierCountry = SupplierCountries.Country, CustomerCountry = CustomerCountries.Country
FROM SupplierCountries
FULL OUTER JOIN CustomerCountries ON SupplierCountries.Country = CustomerCountries.Country
```

Table 57: Displaying records 1 - 25

SupplierCountry	CustomerCountry
NA	Argentina
Australia	NA
NA	Austria
NA	Belgium
Brazil	Brazil
Canada	Canada

SupplierCountry	CustomerCountry
Denmark	Denmark
Finland	Finland
France	France
Germany	Germany
NA	Ireland
Italy	Italy
Japan	NA
NA	Mexico
Netherlands	NA
Norway	Norway
NA	Poland
NA	Portugal
Singapore	NA
Spain	Spain
Sweden	Sweden
NA	Switzerland
UK	UK
USA	USA
NA	Venezuela

48 | Countries with Suppliers or Customers V3

Return country name, total suppliers, and total customers.

```
;WITH CountriesList AS(
  SELECT Distinct(Country)
  FROM Customers
  UNION
  SELECT DISTINCT(Country)
  FROM Suppliers
),
SupplierCountries AS(
SELECT Country, NumCountry = COUNT(*) FROM Suppliers
GROUP BY Country
),
CustomerCountries AS(
SELECT Country, NumCountry = COUNT(*) FROM Customers
GROUP BY Country
)
SELECT CountriesList.Country,
       TotalSuppliers = ISNULL(SupplierCountries.NumCountry, 0),
       TotalCustomers = ISNULL(CustomerCountries.NumCountry, 0)
FROM CountriesList
FULL OUTER JOIN SupplierCountries ON CountriesList.Country = SupplierCountries.Country
FULL OUTER JOIN CustomerCountries ON CountriesList.Country = CustomerCountries.Country
ORDER BY Country
```

Table 58: Displaying records 1 - 25

Country	TotalSuppliers	TotalCustomers
Argentina	0	3

Country	TotalSuppliers	TotalCustomers
Australia	2	0
Austria	0	2
Belgium	0	2
Brazil	1	9
Canada	2	3
Denmark	1	2
Finland	1	2
France	3	11
Germany	3	11
Ireland	0	1
Italy	2	3
Japan	2	0
Mexico	0	5
Netherlands	1	0
Norway	1	1
Poland	0	1
Portugal	0	2
Singapore	1	0
Spain	1	5
Sweden	2	2
Switzerland	0	2
UK	2	7
USA	4	13
Venezuela	0	4

49 | First Order in Each Country

Suppose we wish to know the first order in each country, ordered by order id.

```
;WITH OrdersByCountry AS(
  SELECT ShipCountry, CustomerID, OrderID, OrderDate = CONVERT(DATE, OrderDate),
  RowNumberPerCountry = Row_Number() OVER(PARTITION BY ShipCountry ORDER BY ShipCountry, OrderID)
  FROM Orders
)
SELECT ShipCountry, CustomerID, OrderID, OrderDate
FROM OrdersByCountry
WHERE RowNumberPerCountry = 1
ORDER BY ShipCountry
```

Table 59: 21 records

ShipCountry	CustomerID	OrderID	OrderDate
Argentina	OCEAN	10409	2015-01-09
Austria	ERNSH	10258	2014-07-17
Belgium	SUPRD	10252	2014-07-09
Brazil	HANAR	10250	2014-07-08
Canada	MEREP	10332	2014-10-17
Denmark	SIMOB	10341	2014-10-29
Finland	WARTH	10266	2014-07-26
France	VINET	10248	2014-07-04
Germany	TOMSP	10249	2014-07-05

ShipCountry	CustomerID	OrderID	OrderDate
Ireland	HUNGO	10298	2014-09-05
Italy	MAGAA	10275	2014-08-07
Mexico	CENTC	10259	2014-07-18
Norway	SANTG	10387	2014-12-18
Poland	WOLZA	10374	2014-12-05
Portugal	FURIB	10328	2014-10-14
Spain	ROMEY	10281	2014-08-14
Sweden	FOLKO	10264	2014-07-24
Switzerland	CHOPS	10254	2014-07-11
UK	BSBEV	10289	2014-08-26
USA	RATTC	10262	2014-07-22
Venezuela	HILAA	10257	2014-07-16

50 | Customers with Multiple Orders in 5 Day Period

There are customers for whom freight is a major expense when ordering. By batching up their orders, and making one large order as opposed to multiple smaller orders in a short period of time, they can reduce their freight costs significantly.

Show the customers who have made more than 1 order in a 5 day period. The table shows 25 / 71 rows.

```
SELECT InitialOrder.CustomerID,
       InitialOrderID = InitialOrder.OrderID,
       InitialOrderDate = CONVERT(DATE, InitialOrder.OrderDate),
       NextOrderID = NextOrder.OrderID,
       NextOrderDate = CONVERT(DATE, NextOrder.OrderDate),
       DaysBetweenOrders = DATEDIFF(dd, InitialOrder.OrderDate, NextOrder.OrderDate)
FROM Orders InitialOrder
JOIN Orders NextOrder ON InitialOrder.CustomerID = NextOrder.CustomerID
WHERE InitialOrder.OrderID < NextOrder.OrderID
AND DATEDIFF(dd, InitialOrder.OrderDate, NextOrder.OrderDate) <= 5
ORDER BY InitialOrder.CustomerID, InitialOrder.OrderID
```

Table 60: Displaying records 1 - 25

CustomerID	InitialOrderID	InitialOrderDate	NextOrderID	NextOrderDate	DaysBetweenOrders
ANTON	10677	2015-09-22	10682	2015-09-25	3
AROUT	10741	2015-11-14	10743	2015-11-17	3
BERGS	10278	2014-08-12	10280	2014-08-14	2
BERGS	10444	2015-02-12	10445	2015-02-13	1
BERGS	10866	2016-02-03	10875	2016-02-06	3
BONAP	10730	2015-11-05	10732	2015-11-06	1
BONAP	10871	2016-02-05	10876	2016-02-09	4
BONAP	10932	2016-03-06	10940	2016-03-11	5
BOTTM	10410	2015-01-10	10411	2015-01-10	0
BOTTM	10944	2016-03-12	10949	2016-03-13	1
BOTTM	10975	2016-03-25	10982	2016-03-27	2
BOTTM	11045	2016-04-23	11048	2016-04-24	1
BSBEV	10538	2015-05-15	10539	2015-05-16	1
BSBEV	10943	2016-03-11	10947	2016-03-13	2
EASTC	11047	2016-04-24	11056	2016-04-28	4
ERNSH	10402	2015-01-02	10403	2015-01-03	1

CustomerID	InitialOrderID	InitialOrderDate	NextOrderID	NextOrderDate	DaysBetweenOrders
ERNSH	10771	2015-12-10	10773	2015-12-11	1
ERNSH	10771	2015-12-10	10776	2015-12-15	5
ERNSH	10773	2015-12-11	10776	2015-12-15	4
ERNSH	10968	2016-03-23	10979	2016-03-26	3
ERNSH	11008	2016-04-08	11017	2016-04-13	5
FOLKO	10977	2016-03-26	10980	2016-03-27	1
FOLKO	10980	2016-03-27	10993	2016-04-01	5
FOLKO	10993	2016-04-01	11001	2016-04-06	5
FRANK	10670	2015-09-16	10675	2015-09-19	3

Alternatively, we could do it with a window function. The table below shows 25 / 69 rows.

```
;WITH NextOrderDate AS(
  SELECT CustomerID, OrderDate = CONVERT(DATE, OrderDate),
  NextOrderDate = CONVERT(DATE, LEAD(OrderDate, 1)
  OVER(PARTITION BY CustomerID ORDER BY CustomerID, OrderDate))
  FROM Orders
)
SELECT CustomerID, OrderDate, NextOrderDate, DaysBetweenOrders = DATEDIFF(dd, OrderDate, NextOrderDate)
FROM NextOrderDate
WHERE DATEDIFF(dd, OrderDate, NextOrderDate) <= 5
```

Table 61: Displaying records 1 - 25

CustomerID	OrderDate	NextOrderDate	DaysBetweenOrders
ANTON	2015-09-22	2015-09-25	3
AROUT	2015-11-14	2015-11-17	3
BERGS	2014-08-12	2014-08-14	2
BERGS	2015-02-12	2015-02-13	1
BERGS	2016-02-03	2016-02-06	3
BONAP	2015-11-05	2015-11-06	1
BONAP	2016-02-05	2016-02-09	4
BONAP	2016-03-06	2016-03-11	5
BOTTM	2015-01-10	2015-01-10	0
BOTTM	2016-03-12	2016-03-13	1
BOTTM	2016-03-25	2016-03-27	2
BOTTM	2016-04-23	2016-04-24	1
BSBEV	2015-05-15	2015-05-16	1
BSBEV	2016-03-11	2016-03-13	2
EASTC	2016-04-24	2016-04-28	4
ERNSH	2015-01-02	2015-01-03	1
ERNSH	2015-12-10	2015-12-11	1
ERNSH	2015-12-11	2015-12-15	4
ERNSH	2016-03-23	2016-03-26	3
ERNSH	2016-04-08	2016-04-13	5
FOLKO	2016-03-26	2016-03-27	1
FOLKO	2016-03-27	2016-04-01	5
FOLKO	2016-04-01	2016-04-06	5
FRANK	2015-09-16	2015-09-19	3
GODOS	2016-02-05	2016-02-06	1