Instructor

Mark Bacchus

COMP 1630

TERM PROJECT

By Simon Chen A01010924

July 4, 2017

Contents

[Introduction 1](#_Toc486940116)

[Part A - Database and Tables 2](#_Toc486940117)

[Part B - SQL Statements 8](#_Toc486940118)

[Part C - INSERT, UPDATE, DELETE and VIEWS Statements 18](#_Toc486940119)

[Part D - Stored Procedures and Triggers 27](#_Toc486940120)

[Table Design 35](#_Toc486940121)

[Challenges 37](#_Toc486940122)

[Summary 37](#_Toc486940123)

[Copy of Script 38](#_Toc486940124)

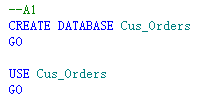
# 

# Introduction

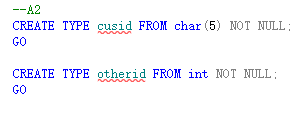
This is a individual term project of COMP1630 summer 2017. The project is about using SQL with Microsoft SQL management studio to accomplish task given by professor Mark Bacchus. By the end of this project, I should be much more familiar with using SQL.

# Part A - Database and Tables

1. Create a database called **Cus\_Orders**.



1. Create a user defined data types for all similar Primary Key attribute columns (e.g. order\_id, product\_id,title\_id**),** to ensure the same data type, length and null ability. See pages 12/13 for specifications.



1. Create the following tables (see column information on pages 12 and 13 ):

Customers

orders

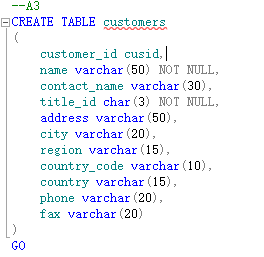
order\_details

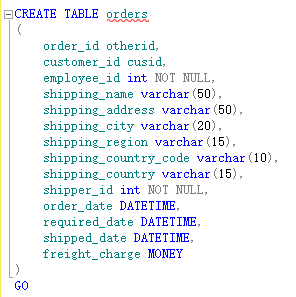
products

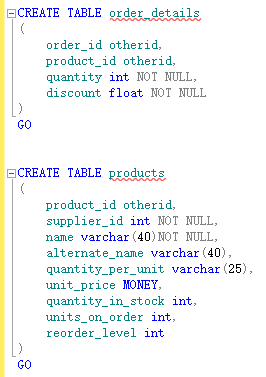
shippers

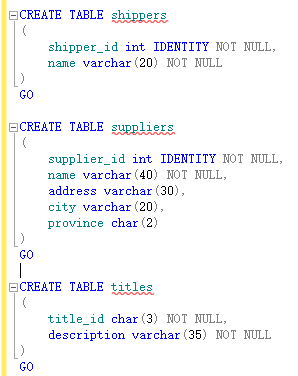
suppliers

titles

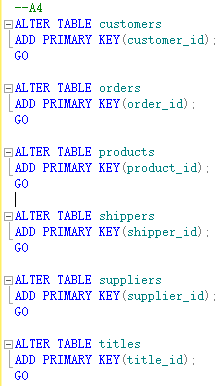




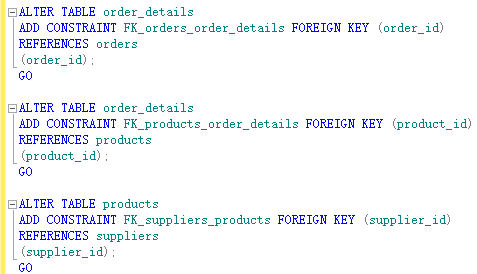




1. Set the **primary** **keys** and **foreign** **keys** for the tables.







1. Set the **constraints** as follows:

**customers** **table** - country should default to Canada

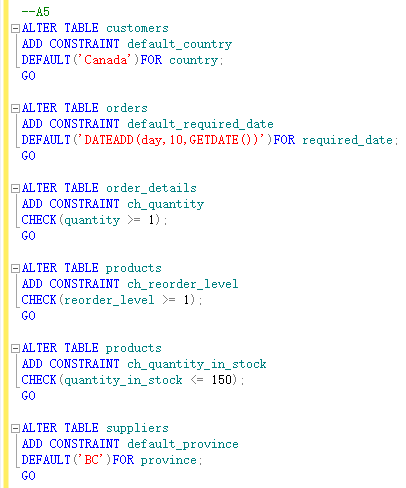
**orders** **table** - required\_date should default to today’s date plus ten days

**order details table** - quantity must be greater than or equal to 1

**products table** - reorder\_level must be greater than or equal to 1

- quantity\_in\_stock value must not be greater than 150

**suppliers table** - province should default to BC



**Note: A complete database diagram will be shown at part C**

1. Load the data into your created tables using the following files:

customers.txt into the customers table (91 rows)

orders.txt into the orders table (1078 rows)

order\_details.txt into the order\_details table (2820 rows)

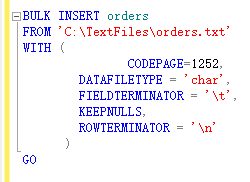
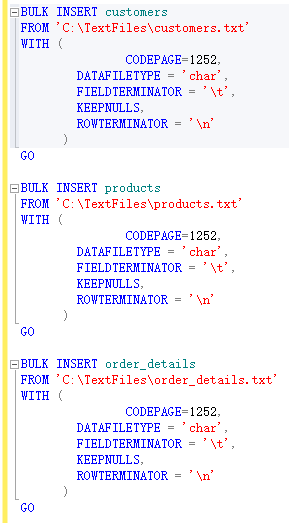
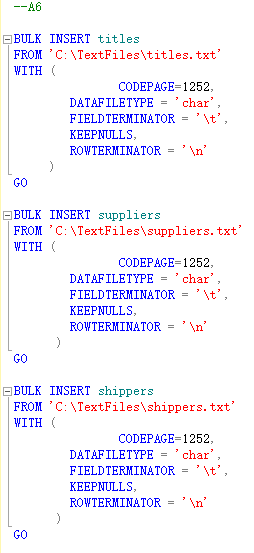
products.txt into the products table (77 rows)

shippers.txt into the shippers table (3 rows)

suppliers.txt into the suppliers table (15 rows)

titles.txt into the titles table (12 rows)

*employees.txt into the employees table which is created in Part C (See Note)*



# Part B - SQL Statements

1. List the customer id, name, city, and country from the customer table. Order the result set by the **customer id**. The query should produce the result set listed below.

customer\_id name city country

--------------- ---------------------------------- ------------- ---------------

ALFKI Alfreds Futterkiste Berlin Germany

ANATR Ana Trujillo Emparedados y helados México D.F. Mexico

ANTON Antonio Moreno Taquería México D.F. Mexico

AROUT Around the Horn London United Kingdom

BERGS Berglunds snabbköp Luleå Sweden

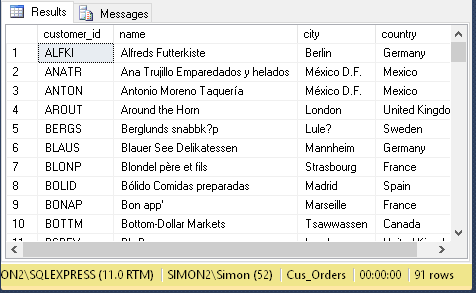
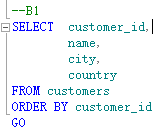
...

WHITC White Clover Markets Seattle United States

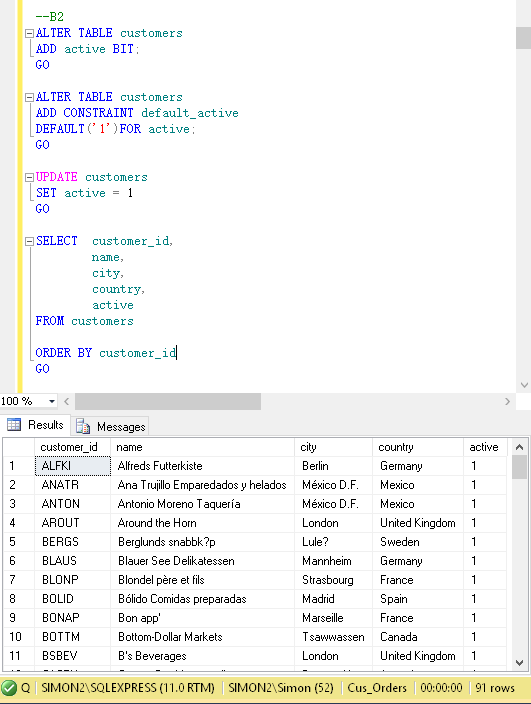
WILMK Wilman Kala Helsinki Finland

WOLZA Wolski Zajazd Warszawa Poland

(91 row(s) affected)



1. Add a new column called **active** to the customers table using the ALTER statement. The only valid values are 1 or 0. The default should be **1**.



1. List all the orders where the order date is between **January 1** and **December 31, 200**1. Display the order id, order date, and a new shipped date calculated by adding 7 days to the shipped date from the orders table, the product name from the product table, the customer name from the customer table, and the cost of the order. Format the date order date and the shipped date as **MON DD YYYY**. Use the formula (quantity \* unit\_price) to calculate the cost of the order. The query should produce the result set listed below.

order\_id product\_name customer\_name order\_date new\_shipped\_date order\_cost

-------- ----------------------------- ---------------- ------------------- --------------------- ----------

10000 Alice Mutton Franchi S.p.A. May 10 2001 May 22 2001 156.0000

10001 NuNuCa Nuß-Nougat-Crème Mère Paillarde May 13 2001 May 30 2001 420.0000

10001 Boston Crab Meat Mère Paillarde May 13 2001 May 30 2001 736.0000

10001 Raclette Courdavault Mère Paillarde May 13 2001 May 30 2001 440.0000

10001 Wimmers gute Semmelknödel Mère Paillarde May 13 2001 May 30 2001 498.7500

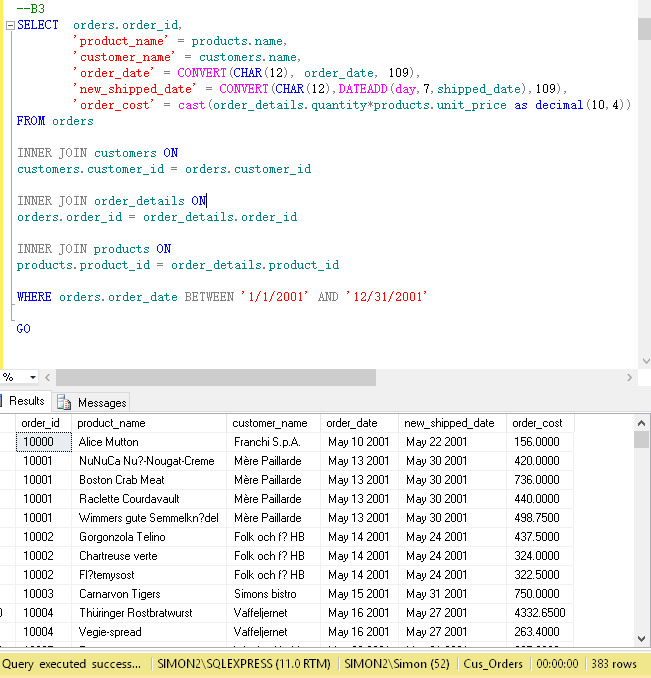
...

10138 Inlagd Sill Du monde entire Dec 27 2001 Jan 10 2002 228.0000

10138 Louisiana Hot Spiced Okra Du monde entire Dec 27 2001 Jan 10 2002 204.0000

10139 Camembert Pierrot Vaffeljernet Dec 30 2001 Jan 16 2002 680.0000

(383 row(s) affected)



1. List all the orders that have **not** been shipped. Display the customer id, name and phone number from the customers table, and the order id and order date from the orders table. Order the result set by the customer name. The query should produce the result set listed below.

customer\_id name phone order\_id order\_date

--------------- --------------------------- -------------- ---------- ---------------------

BLAUS Blauer See Delikatessen 0621-08460 11058 2004-03-23 00:00:00.000

BONAP Bon app' 91.24.45.40 11076 2004-03-30 00:00:00.000

ERNSH Ernst Handel 7675-3425 11008 2004-03-02 00:00:00.000

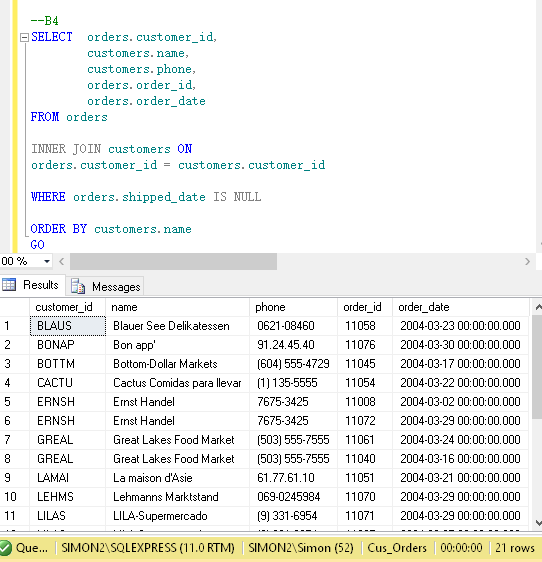
.....

RICAR Ricardo Adocicados (21) 555-3412 11059 2004-03-23 00:00:00.000

RICSU Richter Supermarkt 0897-034214 11075 2004-03-30 00:00:00.000

SIMOB Simons bistro 31 12 34 56 11074 2004-03-30 00:00:00.000

(21 row(s) affected)



1. List all the customers where the region is **NULL**. Display the customer id, name, and city from the customers table, and the title description from the titles table. The query should produce the result set listed below.

customer\_id name city description

-------------- ----------------------------------- --------------- ----------------------

ALFKI Alfreds Futterkiste Berlin Sales Representative

ANATR Ana Trujillo Emparedados y helados México D.F. Owner

ANTON Antonio Moreno Taquería México D.F. Owner

AROUT Around the Horn London Sales Representative

BERGS Berglunds snabbköp Luleå Order Administrator

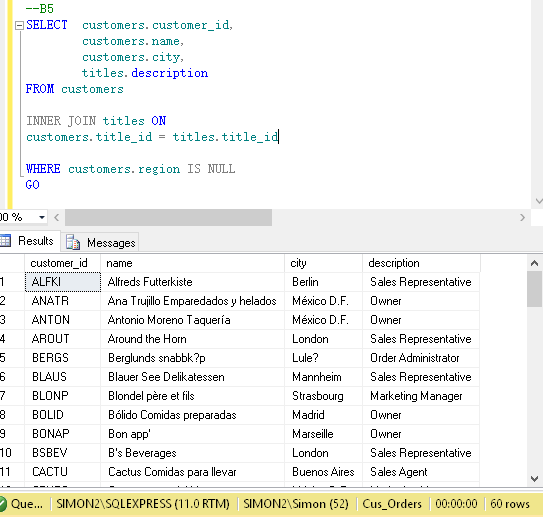
...

WARTH Wartian Herkku Oulu Accounting Manager

WILMK Wilman Kala Helsinki Owner/Marketing Assistant

WOLZA Wolski Zajazd Warszawa Owner

(60 row(s) affected)



1. List the products where the reorder level is **higher than** the quantity in stock. Display the supplier name from the suppliers table, the product name, reorder level, and quantity in stock from the products table. Order the result set by the supplier name. The query should produce the result set listed below.

supplier\_name product\_name reorder\_level quantity\_in\_stock

------------------------ --------------------- ---------------- -----------------

Armstrong Company Queso Cabrales 30 22

Cadbury Products Ltd. Ipoh Coffee 25 17

Cadbury Products Ltd. Røgede sild 15 5

Campbell Company Gnocchi di nonna Alice 30 21

Dare Manufacturer Ltd. Scottish Longbreads 15 6

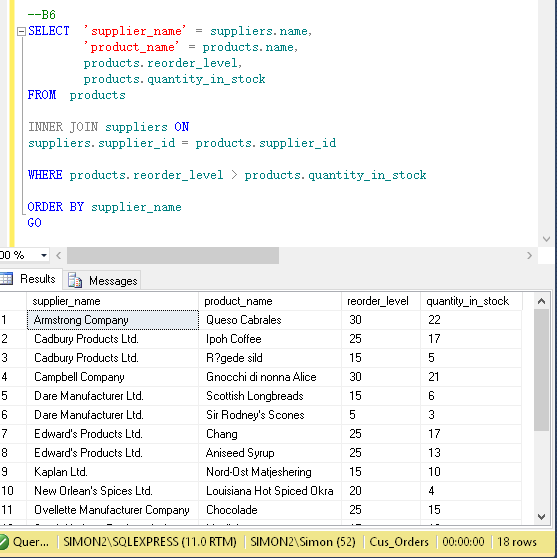
...

Steveston Export Company Gravad lax 25 11

Steveston Export Company Outback Lager 30 15

Yves Delorme Ltd. Longlife Tofu 5 4

(18 row(s) affected)



1. Calculate the length in years from **January 1, 2008** and when an order was shipped where the shipped date is **not null**. Display the order id, and the shipped date from the orders table, the customer name, and the contact name from the customers table, and the length in years for each order. Display the shipped date in the format MMM DD YYYY. Order the result set by order id and the calculated years. The query should produce the result set listed below.

order\_id name contact\_name shipped\_date elapsed

----------- --------------------------- --------------------------- ------------ -------

10000 Franchi S.p.A. Paolo Accorti May 15 2001 7

10001 Mère Paillarde Jean Fresnière May 23 2001 7

10002 Folk och fä HB Maria Larsson May 17 2001 7

10003 Simons bistro Jytte Petersen May 24 2001 7

10004 Vaffeljernet Palle Ibsen May 20 2001 7

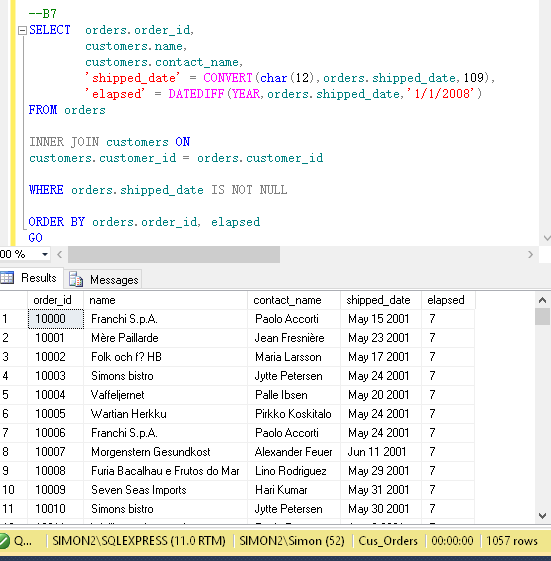
...

11066 White Clover Markets Karl Jablonski Mar 28 2004 4

11067 Drachenblut Delikatessen Sven Ottlieb Mar 30 2004 4

11069 Tortuga Restaurante Miguel Angel Paolino Mar 30 2004 4

(1057 row(s) affected)



1. List number of customers with names beginning with each letter of the alphabet. Ignore customers whose name begins with the letter **S**. Do not display the letter and count unless **at least two** customer’s names begin with the letter. The query should produce the result set listed below.

name total

------ -----------

A 4

B 7

C 5

D 3

E 2

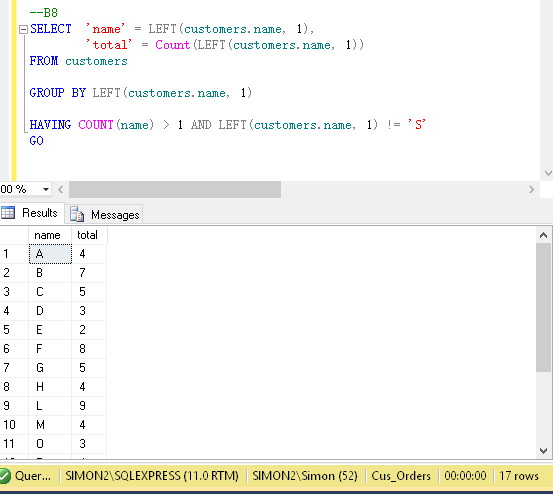
...

T 6

V 3

W 5

(17 row(s) affected)



1. List the order details where the quantity is **greater than 100**. Display the order id and quantity from the order\_details table, the product id and reorder level from the products table, and the supplier id from the suppliers table. Order the result set by the order id. The query should produce the result set listed below.

order\_id quantity product\_id reorder\_level supplier\_id

----------- ----------- -------------- ----------------- -----------

10193 110 43 25 10

10226 110 29 0 12

10398 120 55 20 15

10451 120 55 20 15

10515 120 27 30 11

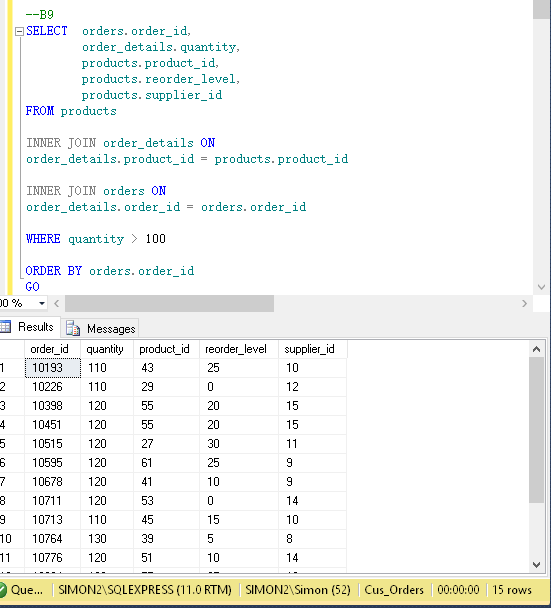
...

10895 110 24 0 10

11017 110 59 0 8

11072 130 64 30 12

(15 row(s) affected)



1. List the products which contain **tofu** or **chef** in their name. Display the product id, product name, quantity per unit and unit price from the products table. Order the result set by product name. The query should produce the result set listed below.

product\_id name quantity\_per\_unit unit\_price

-------------- ------------------------------------- ---------------------- --------------

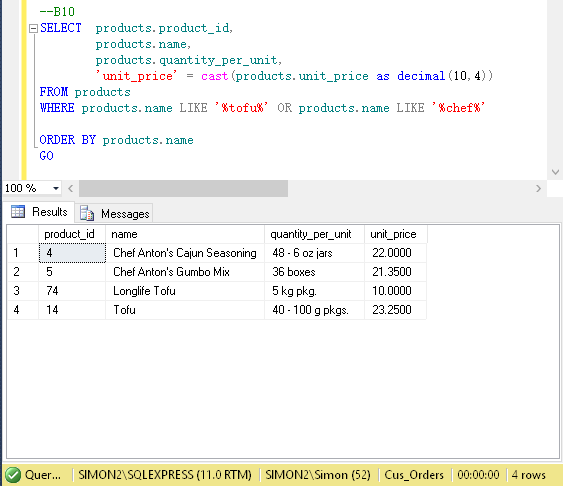
4 Chef Anton's Cajun Seasoning 48 - 6 oz jars 22.0000

5 Chef Anton's Gumbo Mix 36 boxes 21.3500

74 Longlife Tofu 5 kg pkg. 10.0000

14 Tofu 40 - 100 g pkgs. 23.2500

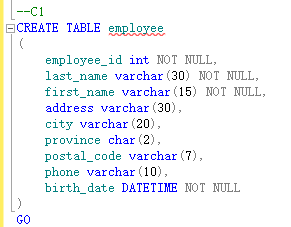
(4 row(s) affected)



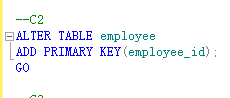
# Part C - INSERT, UPDATE, DELETE and VIEWS Statements

1. Create an **employee** table with the following columns:

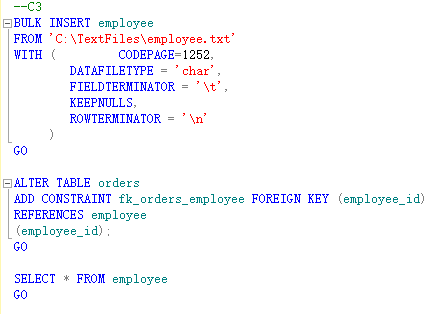
|  |  |  |  |
| --- | --- | --- | --- |
| *Column Name* | *Data Type* | *Length* | *Null Values* |
| employee\_id | int |  | No |
| last\_name | varchar | 30 | No |
| first\_name | varchar | 15 | No |
| address | varchar | 30 |  |
| city | varchar | 20 |  |
| province | char | 2 |  |
| postal\_code | varchar | 7 |  |
| phone | varchar | 10 |  |
| birth\_date | datetime |  | No |

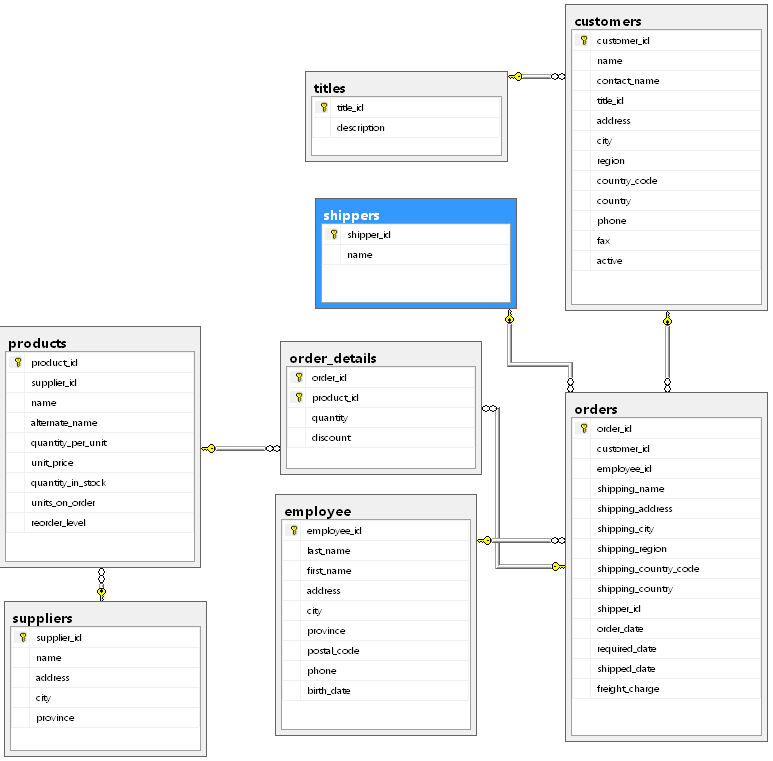


1. The **primary key** for the employee table should be the employee id.

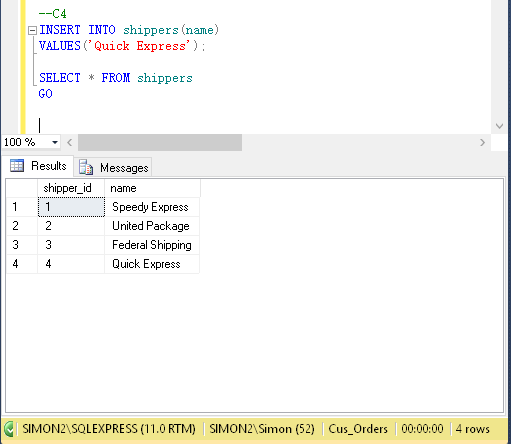


1. Load the data into the employee table using the employee.txt file; **9** rows. In addition, **create the relationship** to enforce referential integrity between the employee and orders tables.

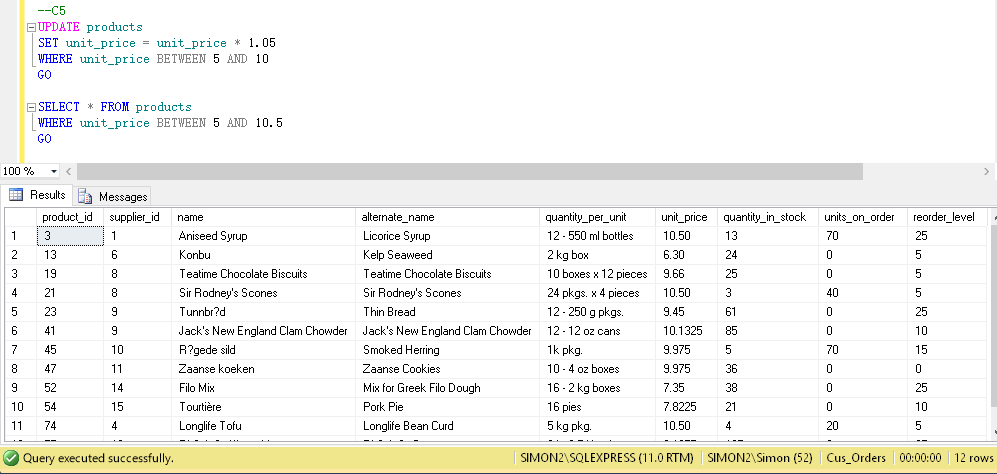




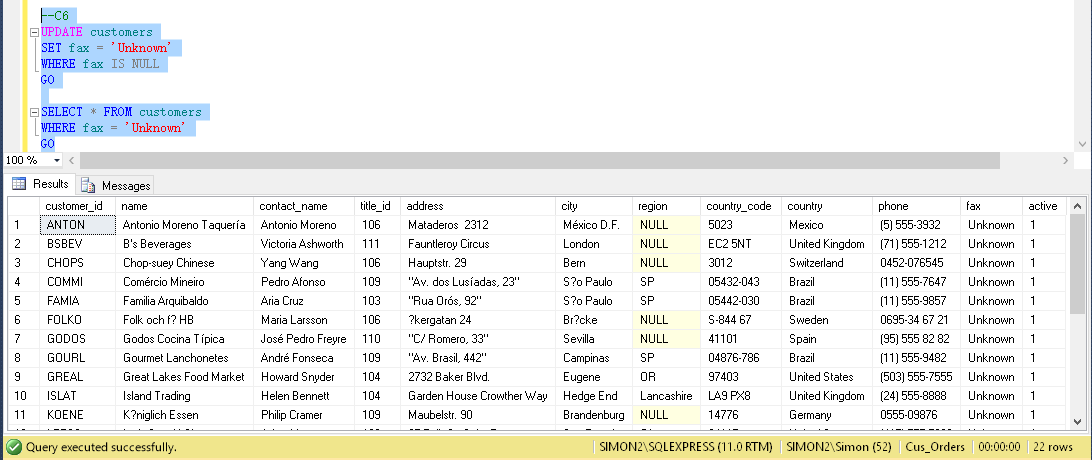
1. Using the INSERT statement, add the shipper **Quick Express** to the shippers table.



1. Using the UPDATE statement, increate the unit price in the products table of all rows with a current unit price between **$5.00** and **$10.00** by **5%**; 12 rows affected.



1. Using the UPDATE statement, change the fax value to **Unknown** for all rows in the customers table where the current fax value is **NULL**; 22 rows affected.



1. Create a view called **vw\_order\_cost** to list the cost of the orders. Display the order id and order\_date from the orders table, the product id from the products table, the customer name from the customers tble, and the order cost. To calculate the cost of the orders, use the formula (order\_details.quantity \* products.unit\_price). Run the view for the order ids between **10000** and **10200**. The view should produce the result set listed below.

order\_id order\_date product\_id name order\_cost

--------- ------------------------------ -------------- ---------------------------------- --------------

10000 2001-05-10 00:00:00.000 17 Franchi S.p.A. 156.0000

10001 2001-05-13 00:00:00.000 25 Mère Paillarde 420.0000

10001 2001-05-13 00:00:00.000 40 Mère Paillarde 736.0000

10001 2001-05-13 00:00:00.000 59 Mère Paillarde 440.0000

10001 2001-05-13 00:00:00.000 64 Mère Paillarde 498.7500

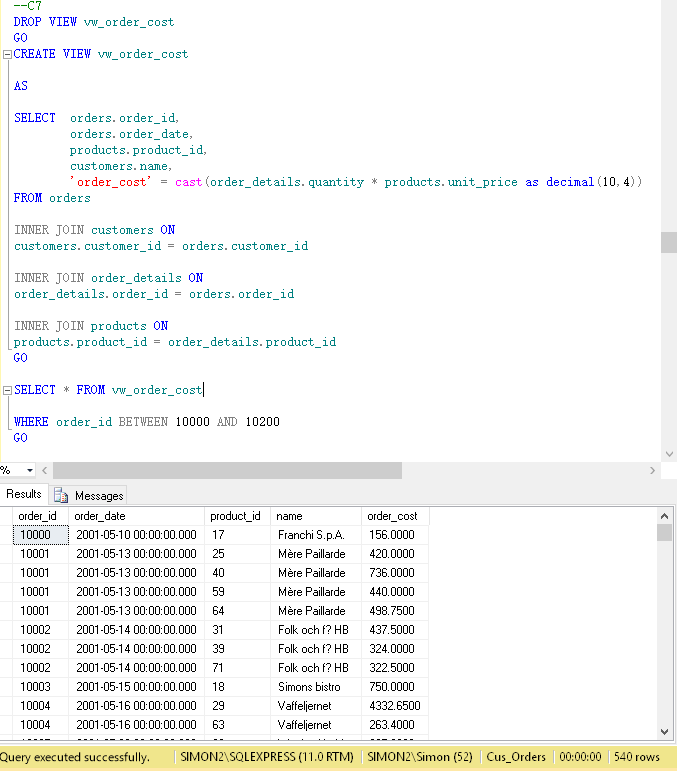
...

10199 2002-03-27 00:00:00.000 3 Save-a-lot Markets 400.0000

10199 2002-03-27 00:00:00.000 39 Save-a-lot Markets 720.0000

10200 2002-03-30 00:00:00.000 11 Bólido Comidas preparadas 588.0000

(540 row(s) affected)



1. Create a view called **vw\_list\_employees** to list all the employees and all the columns in the employee table. Run the view for employee ids **5**, **7**, and **9**. Display the employee id, last name, first name, and birth date. Format the name as last name followed by a comma and a space followed by the first name. Format the birth date as **YYYY.MM.DD**. The view should produce the result set listed below.

employee\_id name birth\_date

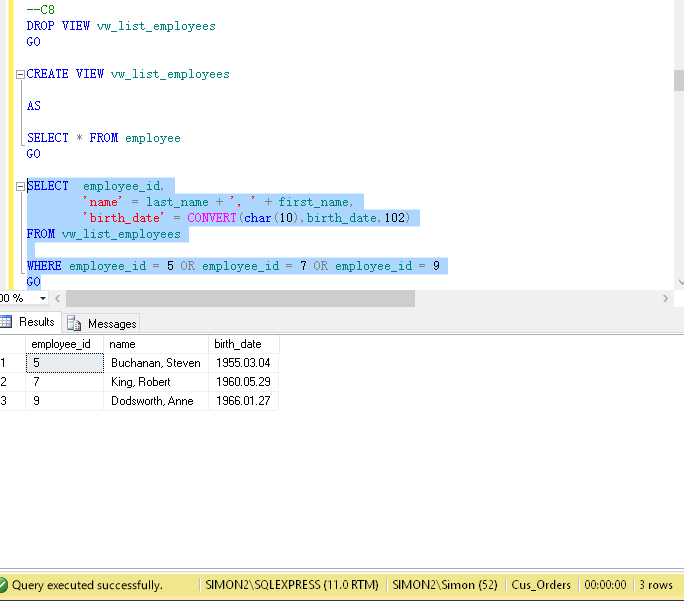
---------------- -------------------------------- ------------

5 Buchanan, Steven 1955.03.04

7 King, Robert 1960.05.29

9 Dodsworth, Anne 1966.01.27

(3 row(s) affected)



1. Create a view called **vw\_all\_orders** to list all the orders. Display the order id and shipped date from the orders table, and the customer id, name, city, and country from the customers table. Run the view for orders shipped from **January 1, 2002** and **December 31, 2002**, formatting the shipped date as **MON DD YYYY**. Order the result set by customer name and country. The view should produce the result set listed below.

order\_id customer\_id customer\_name city country shipped\_date

---------- --------------- ------------------------------------------- ---------------- --------------- ------------

10308 ANATR Ana Trujillo Emparedados y helados México D.F. Mexico Aug 18 2002

10365 ANTON Antonio Moreno Taquería México D.F. Mexico Oct 26 2002

10137 ANTON Antonio Moreno Taquería México D.F. Mexico Jan 22 2002

10142 ANTON Antonio Moreno Taquería México D.F. Mexico Jan 8 2002

10218 ANTON Antonio Moreno Taquería México D.F. Mexico May 25 2002

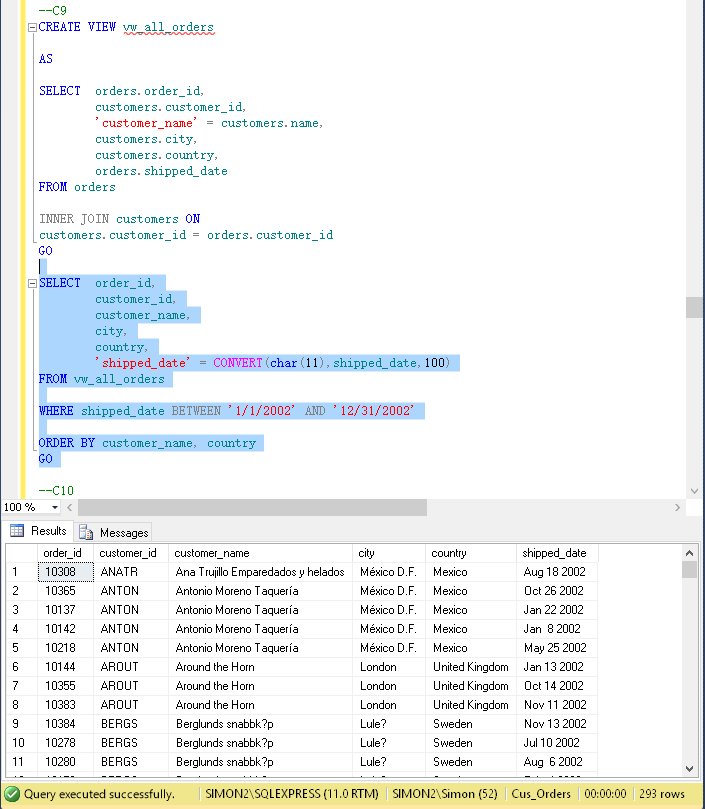
...

10344 WHITC White Clover Markets Seattle United States Sep 29 2002

10269 WHITC White Clover Markets Seattle United States Jul 3 2002

10374 WOLZA Wolski Zajazd Warszawa Poland Nov 2 2002

(293 row(s) affected)



1. Create a view listing the suppliers and the items they have shipped. Display the supplier id and name from the suppliers table, and the product id and name from the products table. Run the view. The view should produce the result set listed below, *although not necessarily in the same order.*

supplier\_id supplier\_name product\_id product\_name

--------------- ---------------------------------------- -------------- ------------------------------

9 Silver Spring Wholesale Market 23 Tunnbröd

11 Ovellette Manufacturer Company 46 Spegesild

15 Campbell Company 69 Gudbrandsdalsost

12 South Harbour Products Ltd. 77 Original Frankfurter grüne Soße

14 St. Jean's Company 31 Gorgonzola Telino

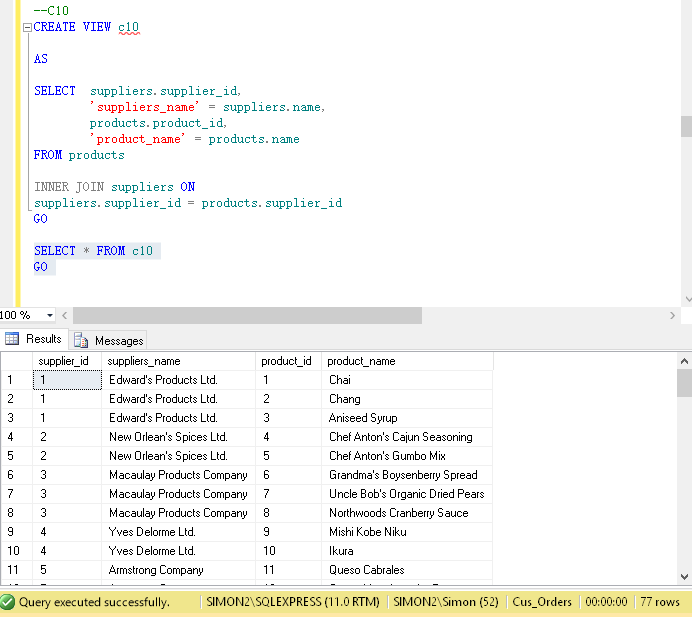
...

7 Steveston Export Company 63 Vegie-spread

3 Macaulay Products Company 8 Northwoods Cranberry Sauce

15 Campbell Company 55 Pâté chinois

(77 row(s) affected)



# Part D - Stored Procedures and Triggers

1. Create a stored procedure called **sp\_customer\_city** displaying the customers living in a particular city. The **city** will be an **input parameter** for the stored procedure. Display the customer id, name, address, city and phone from the customers table. Run the stored procedure displaying customers living in **London**. The stored procedure should produce the result set listed below.

customer\_id name address city phone

--------------- ------------------------ ---------- ------------------------------------- ------ ------------

AROUT Around the Horn 120 Hanover Sq. London (71) 555-7788

BSBEV B's Beverages Fauntleroy Circus London (71) 555-1212

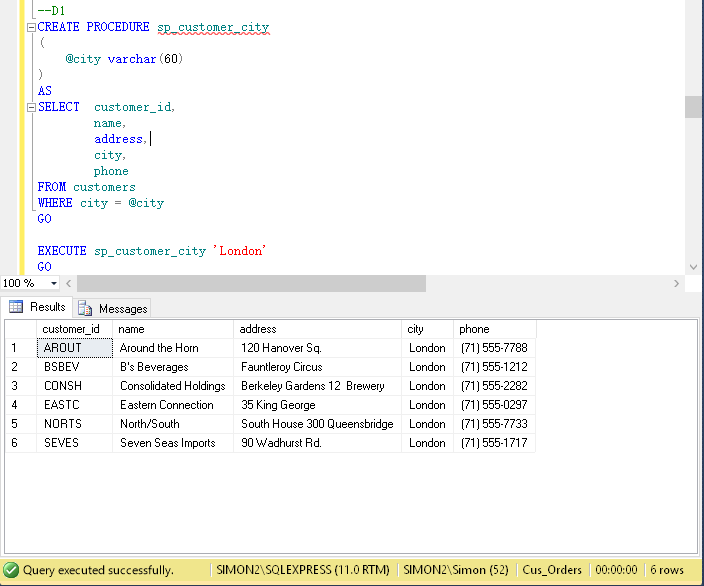
CONSH Consolidated Holdings Berkeley Gardens 12 Brewery London (71) 555-2282

EASTC Eastern Connection 35 King George London (71) 555-0297

NORTS North/South South House 300 Queensbridge London (71) 555-7733

SEVES Seven Seas Imports 90 Wadhurst Rd. London (71) 555-1717

(6 row(s) affected)



1. Create a stored procedure called **sp\_orders\_by\_dates** displaying the orders shipped between particular dates. The **start** and **end** date will be **input parameters** for the stored procedure. Display the order id, customer id, and shipped date from the orders table, the customer name from the customer table, and the shipper name from the shippers table. Run the stored procedure displaying orders from **January 1, 2003** to **June 30, 2003**. The stored procedure should produce the result set listed below.

order\_id customer\_id customer\_name shipper\_name shipped\_date

---------- --------------- ------------------------------------- -------------------- ----------------------

10423 GOURL Gourmet Lanchonetes Federal Shipping 2003-01-18 00:00:00.000

10425 LAMAI La maison d'Asie United Package 2003-01-08 00:00:00.000

10427 PICCO Piccolo und mehr United Package 2003-01-25 00:00:00.000

10429 HUNGO Hungry Owl All-Night Grocers United Package 2003-01-01 00:00:00.000

10431 BOTTM Bottom-Dollar Markets United Package 2003-01-01 00:00:00.000

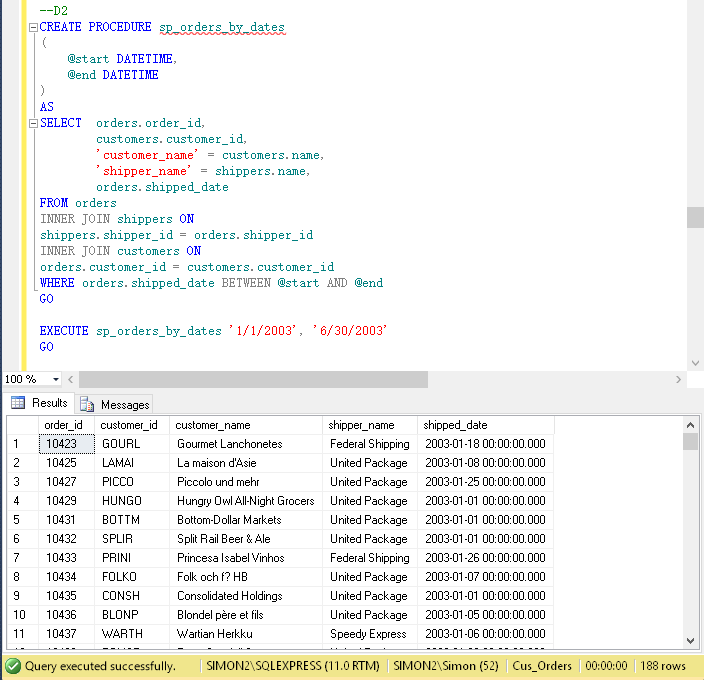
...

10615 WILMK Wilman Kala Federal Shipping 2003-06-30 00:00:00.000

10616 GREAL Great Lakes Food Market United Package 2003-06-29 00:00:00.000

10617 GREAL Great Lakes Food Market United Package 2003-06-28 00:00:00.000

(188 row(s) affected)



1. Create a stored procedure called **sp\_product\_listing** listing a specified product ordered during a specified month and year. The **product** and the **month** and **year** will be **input** **parameters** for the stored procedure. Display the product name, unit price, and quantity in stock from the products table, and the supplier name from the suppliers table. Run the stored procedure displaying a product name containing **Jack** and the month of the order date is **June** and the year is **2001**. The stored procedure should produce the result set listed below.

product\_name unit\_price quantity\_in\_stock supplier\_name

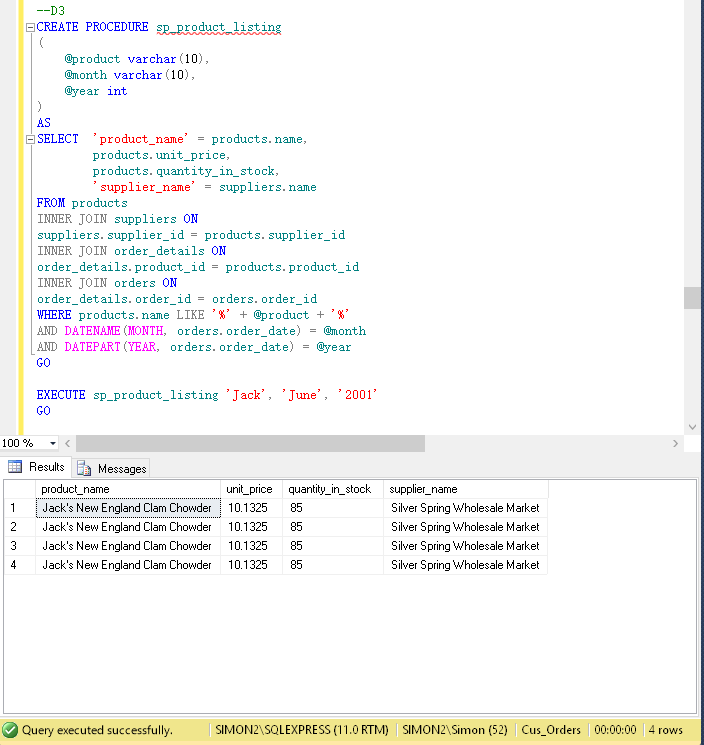
--------------------------------------------- ------------- --------------------- -----------------------------------

Jack's New England Clam Chowder 10.1325 85 Silver Spring Wholesale Market

Jack's New England Clam Chowder 10.1325 85 Silver Spring Wholesale Market

Jack's New England Clam Chowder 10.1325 85 Silver Spring Wholesale Market

Jack's New England Clam Chowder 10.1325 85 Silver Spring Wholesale Market

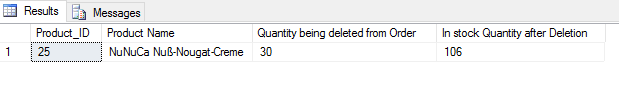
(4 row(s) affect

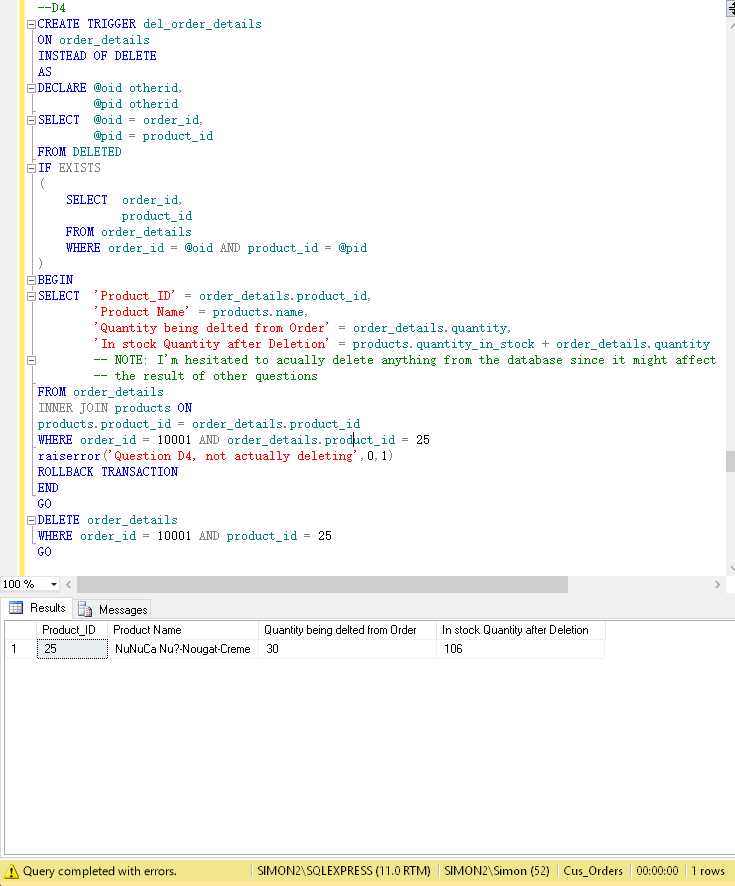
1. Create a **DELETE** trigger on the order\_details table to display the information shown below when you issue the following statement:

DELETE order\_details

WHERE order\_id=10001 AND product\_id=25

You should get the following results:





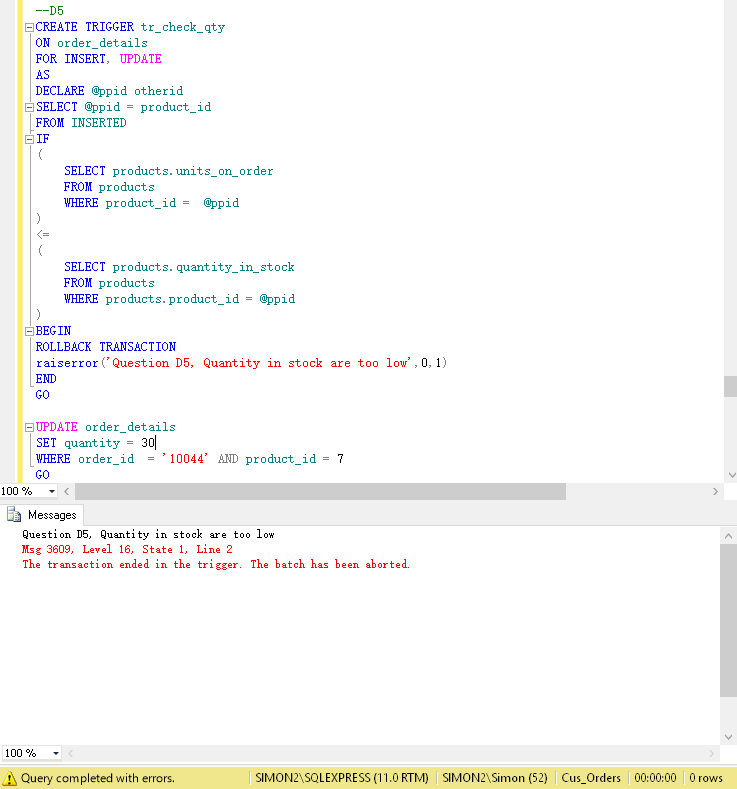
1. Create an **INSERT** and **UPDATE** trigger called **tr\_check\_qty** on the order\_details table to only allow orders of products that have a quantity in stock greater than or equal to the units ordered. Run the following query to verify your trigger.

UPDATE order\_details

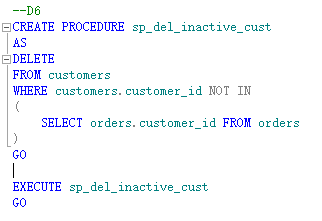
SET quantity = 30

WHERE order\_id = '10044'

AND product\_id = 7



1. Create a stored procedure called **sp\_del\_inactive\_cust** to **delete** customers that have no orders. The stored procedure should delete **1** row.



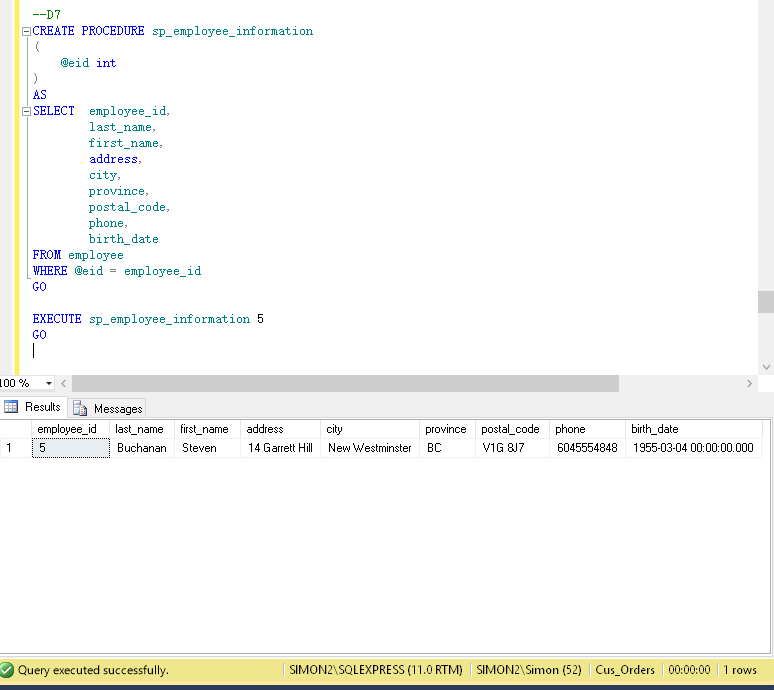
1. Create a stored procedure called **sp\_employee\_information** to display the employee information for a particular employee. The **employee id** will be an **input** **parameter** for the stored procedure. Run the stored procedure displaying information for employee id of **5**. The stored procedure should produce the result set listed below.

employee\_id last\_name first\_name address city province postal\_code phone birth\_date

--------------- ------------- ------------- ----------------- --------------------- ---------- -------------- -------------- -------------------------------

5 Buchanan Steven 14 Garrett Hill New Westminster BC V1G 8J7 6045554848 1955-03-04 00:00:00.000

(1 row(s) affected)



1. Create a stored procedure called **sp\_reorder\_qty** to show when the reorder level subtracted from the quantity in stock is less than a specified value. The **unit** value will be an **input** **parameter** for the stored procedure. Display the product id, quantity in stock, and reorder level from the products table, and the supplier name, address, city, and province from the suppliers table. Run the stored procedure displaying the information for a value of **5**. The stored procedure should produce the result set listed below.

product\_id name address city province qty reorder\_level

-------------- ---------------------------------- ------------------------ ------------------ -------- --- -------------

2 Edward's Products Ltd. 1125 Howe Street Vancouver BC 17 25

3 Edward's Products Ltd. 1125 Howe Street Vancouver BC 13 25

5 New Orlean's Spices Ltd. 1040 Georgia Street West Vancouver BC 0 0

11 Armstrong Company 1638 Derwent Way Richmond BC 22 30

17 Steveston Export Company 2951 Moncton Street Richmond BC 0 0

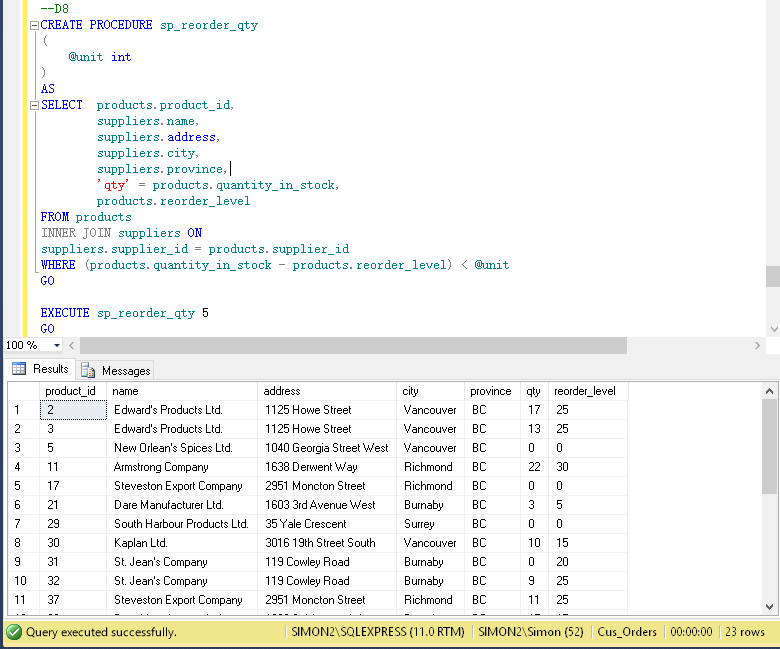
...

68 Dare Manufacturer Ltd. 1603 3rd Avenue West Burnaby BC 6 15

70 Steveston Export Company 2951 Moncton Street Richmond BC 15 30

74 Yves Delorme Ltd. 3050 Granville Street New Westminster BC 4 5

(23 row(s) affected)



1. Create a stored procedure called **sp\_unit\_prices** for the product table where the **unit price** is **between particular values**. The **two unit prices** will be **input** **parameters** for the stored procedure. Display the product id, product name, alternate name, and unit price from the products table. Run the stored procedure to display products where the unit price is between **$5.00** and **$10.00**. The stored procedure should produce the result set listed below.

product\_id name alternate\_name unit\_price

-------------- --------------------------------- -------------------------------- --------------

13 Konbu Kelp Seaweed 6.30

19 Teatime Chocolate Biscuits Teatime Chocolate Biscuits 9.66

23 Tunnbr÷d Thin Bread 9.45

45 R°gede sild Smoked Herring 9.975

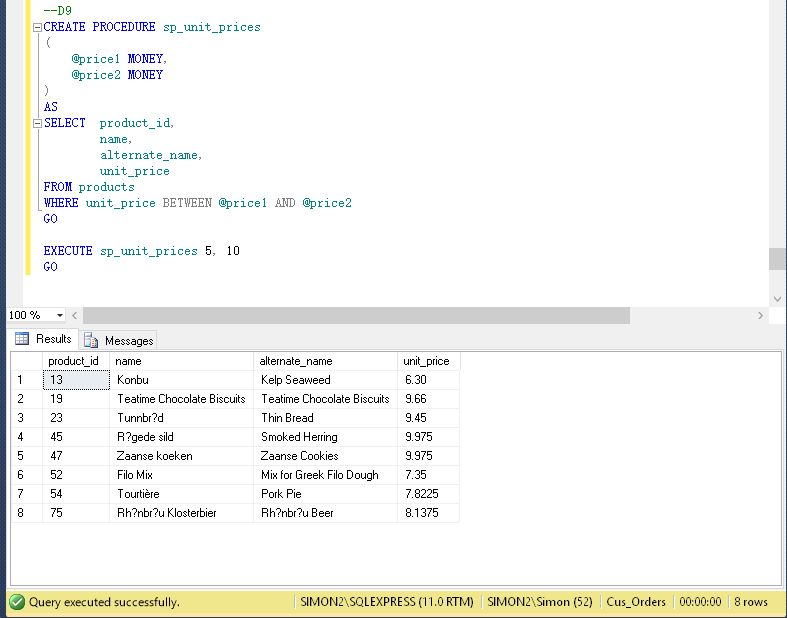
47 Zaanse koeken Zaanse Cookies 9.975

52 Filo Mix Mix for Greek Filo Dough 7.35

54 TourtiÞre Pork Pie 7.8225

75 Rh÷nbrõu Klosterbier Rh÷nbrõu Beer 8.1375

1. row(s) affected)



# Table Design

**customers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Column Name* | *Data Type* | *Length* |  | *Null Values* |
| customer\_id | char | 5 | User-Defined Data Type | No |
| name | varchar | 50 |  | No |
| contact\_name | varchar | 30 |  |  |
| title\_id | char | 3 |  | No |
| address | varchar | 50 |  |  |
| city | varchar | 20 |  |  |
| region | varchar | 15 |  |  |
| country\_code | varchar | 10 |  |  |
| country | varchar | 15 |  |  |
| phone | varchar | 20 |  |  |
| fax | varchar | 20 |  |  |

**orders**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Column Name* | *Data Type* | *Length* |  | *Null Values* |
| order\_id | int |  | User-Defined Data Type | No |
| customer\_id | char | 5 | User-Defined Data Type | No |
| employee\_id | int |  |  | No |
| shipping\_name | varchar | 50 |  |  |
| shipping\_address | varchar | 50 |  |  |
| shipping\_city | varchar | 20 |  |  |
| shipping\_region | varchar | 15 |  |  |
| shipping\_country\_code | varchar | 10 |  |  |
| shipping\_country | varchar | 15 |  |  |
| shipper\_id | int |  |  | No |
| order\_date | datetime |  |  |  |
| required\_date | datetime |  |  |  |
| shipped\_date | datetime |  |  |  |
| freight\_charge | money |  |  |  |

**order\_details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Column Name* | *Data Type* | *Length* |  | *Null Values* |
| order\_id | int |  | User-Defined Data Type | No |
| product\_id | int |  | User-Defined Data Type | No |
| quantity | int |  |  | No |
| discount | float |  |  | No |

**products**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Column Name* | *Data Type* | *Length* |  | *Null Values* |
| product\_id | int |  | User-Defined Data Type | No |
| supplier\_id | int |  |  | No |
| name | varchar | 40 |  | No |
| alternate\_name | varchar | 40 |  |  |
| quantity\_per\_unit | varchar | 25 |  |  |
| unit\_price | money |  |  |  |
| quantity\_in\_stock | int |  |  |  |
| units\_on\_order | int |  |  |  |
| reorder\_level | int |  |  |  |

**shippers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Column Name* | *Data Type* | *Length* |  | *Null Values* |
| shipper\_id | int |  | IDENTITY | No |
| name | varchar | 20 |  | No |

**suppliers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Column Name* | *Data Type* | *Length* |  | *Null Values* |
| supplier\_id | int |  | IDENTITY | No |
| name | varchar | 40 |  | No |
| address | varchar | 30 |  |  |
| city | varchar | 20 |  |  |
| province | char | 2 |  |  |

**titles**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Column Name* | *Data Type* | *Length* |  | *Null Values* |
| title\_id | char | 3 |  | No |
| description | varchar | 35 |  | No |

# 

# Challenges

Part D question 4, I’m not sure whether I should delete from the database since by doing so, it might affect the result of other questions. I would also have to update the database as well which the question didn’t specify.

# 

# Summary

I’ve learned a lot throughout this project; When I first started, I didn’t even know how to use different values from different tables. However, as the project progress, I’m getting better and better at problem solving.

# Copy of Script

-- PART A

USE master

GO

IF exists (SELECT \* FROM sysdatabases WHERE name='Cus\_Orders')

BEGIN

raiserror('Dropping existing Cus\_Orders database ....',0,1)

DROP database Cus\_Orders

END

GO

--A1

CREATE DATABASE Cus\_Orders

GO

USE Cus\_Orders

GO

--A2

CREATE TYPE cusid FROM char(5) NOT NULL;

GO

CREATE TYPE otherid FROM int NOT NULL;

GO

--A3

CREATE TABLE customers

(

customer\_id cusid,

name varchar(50) NOT NULL,

contact\_name varchar(30),

title\_id char(3) NOT NULL,

address varchar(50),

city varchar(20),

region varchar(15),

country\_code varchar(10),

country varchar(15),

phone varchar(20),

fax varchar(20)

)

GO

CREATE TABLE orders

(

order\_id otherid,

customer\_id cusid,

employee\_id int NOT NULL,

shipping\_name varchar(50),

shipping\_address varchar(50),

shipping\_city varchar(20),

shipping\_region varchar(15),

shipping\_country\_code varchar(10),

shipping\_country varchar(15),

shipper\_id int NOT NULL,

order\_date DATETIME,

required\_date DATETIME,

shipped\_date DATETIME,

freight\_charge MONEY

)

GO

CREATE TABLE order\_details

(

order\_id otherid,

product\_id otherid,

quantity int NOT NULL,

discount float NOT NULL

)

GO

CREATE TABLE products

(

product\_id otherid,

supplier\_id int NOT NULL,

name varchar(40)NOT NULL,

alternate\_name varchar(40),

quantity\_per\_unit varchar(25),

unit\_price MONEY,

quantity\_in\_stock int,

units\_on\_order int,

reorder\_level int

)

GO

CREATE TABLE shippers

(

shipper\_id int IDENTITY NOT NULL,

name varchar(20) NOT NULL

)

GO

CREATE TABLE suppliers

(

supplier\_id int IDENTITY NOT NULL,

name varchar(40) NOT NULL,

address varchar(30),

city varchar(20),

province char(2)

)

GO

CREATE TABLE titles

(

title\_id char(3) NOT NULL,

description varchar(35) NOT NULL

)

GO

--A4

ALTER TABLE customers

ADD PRIMARY KEY(customer\_id);

GO

ALTER TABLE orders

ADD PRIMARY KEY(order\_id);

GO

ALTER TABLE products

ADD PRIMARY KEY(product\_id);

GO

ALTER TABLE shippers

ADD PRIMARY KEY(shipper\_id);

GO

ALTER TABLE suppliers

ADD PRIMARY KEY(supplier\_id);

GO

ALTER TABLE titles

ADD PRIMARY KEY(title\_id);

GO

ALTER TABLE order\_details

ADD PRIMARY KEY(order\_id,product\_id);

GO

-- fk

ALTER TABLE orders

ADD CONSTRAINT FK\_customers\_orders FOREIGN KEY (customer\_id)

REFERENCES customers

(customer\_id);

GO

ALTER TABLE customers

ADD CONSTRAINT FK\_customers\_titles FOREIGN KEY (title\_id)

REFERENCES titles

(title\_id);

GO

ALTER TABLE orders

ADD CONSTRAINT FK\_shippers\_orders FOREIGN KEY (shipper\_id)

REFERENCES shippers

(shipper\_id);

GO

ALTER TABLE order\_details

ADD CONSTRAINT FK\_orders\_order\_details FOREIGN KEY (order\_id)

REFERENCES orders

(order\_id);

GO

ALTER TABLE order\_details

ADD CONSTRAINT FK\_products\_order\_details FOREIGN KEY (product\_id)

REFERENCES products

(product\_id);

GO

ALTER TABLE products

ADD CONSTRAINT FK\_suppliers\_products FOREIGN KEY (supplier\_id)

REFERENCES suppliers

(supplier\_id);

GO

--A5

ALTER TABLE customers

ADD CONSTRAINT default\_country

DEFAULT('Canada')FOR country;

GO

ALTER TABLE orders

ADD CONSTRAINT default\_required\_date

DEFAULT('DATEADD(day,10,GETDATE())')FOR required\_date;

GO

ALTER TABLE order\_details

ADD CONSTRAINT ch\_quantity

CHECK(quantity >= 1);

GO

ALTER TABLE products

ADD CONSTRAINT ch\_reorder\_level

CHECK(reorder\_level >= 1);

GO

ALTER TABLE products

ADD CONSTRAINT ch\_quantity\_in\_stock

CHECK(quantity\_in\_stock <= 150);

GO

ALTER TABLE suppliers

ADD CONSTRAINT default\_province

DEFAULT('BC')FOR province;

GO

--A6

BULK INSERT titles

FROM 'C:\TextFiles\titles.txt'

WITH (

CODEPAGE=1252,

DATAFILETYPE = 'char',

FIELDTERMINATOR = '\t',

KEEPNULLS,

ROWTERMINATOR = '\n'

)

GO

BULK INSERT suppliers

FROM 'C:\TextFiles\suppliers.txt'

WITH (

CODEPAGE=1252,

DATAFILETYPE = 'char',

FIELDTERMINATOR = '\t',

KEEPNULLS,

ROWTERMINATOR = '\n'

)

GO

BULK INSERT shippers

FROM 'C:\TextFiles\shippers.txt'

WITH (

CODEPAGE=1252,

DATAFILETYPE = 'char',

FIELDTERMINATOR = '\t',

KEEPNULLS,

ROWTERMINATOR = '\n'

)

GO

BULK INSERT customers

FROM 'C:\TextFiles\customers.txt'

WITH (

CODEPAGE=1252,

DATAFILETYPE = 'char',

FIELDTERMINATOR = '\t',

KEEPNULLS,

ROWTERMINATOR = '\n'

)

GO

BULK INSERT products

FROM 'C:\TextFiles\products.txt'

WITH (

CODEPAGE=1252,

DATAFILETYPE = 'char',

FIELDTERMINATOR = '\t',

KEEPNULLS,

ROWTERMINATOR = '\n'

)

GO

BULK INSERT order\_details

FROM 'C:\TextFiles\order\_details.txt'

WITH (

CODEPAGE=1252,

DATAFILETYPE = 'char',

FIELDTERMINATOR = '\t',

KEEPNULLS,

ROWTERMINATOR = '\n'

)

GO

BULK INSERT orders

FROM 'C:\TextFiles\orders.txt'

WITH (

CODEPAGE=1252,

DATAFILETYPE = 'char',

FIELDTERMINATOR = '\t',

KEEPNULLS,

ROWTERMINATOR = '\n'

)

GO

--PART B

--B1

SELECT customer\_id,

name,

city,

country

FROM customers

ORDER BY customer\_id

GO

--B2

ALTER TABLE customers

ADD active BIT;

GO

ALTER TABLE customers

ADD CONSTRAINT default\_active

DEFAULT('1')FOR active;

GO

UPDATE customers

SET active = 1

GO

SELECT customer\_id,

name,

city,

country,

active

FROM customers

ORDER BY customer\_id

GO

--B3

SELECT orders.order\_id,

'product\_name' = products.name,

'customer\_name' = customers.name,

'order\_date' = CONVERT(CHAR(12), order\_date, 109),

'new\_shipped\_date' = CONVERT(CHAR(12),DATEADD(day,7,shipped\_date),109),

'order\_cost' = cast(order\_details.quantity\*products.unit\_price as decimal(10,4))

FROM orders

INNER JOIN customers ON

customers.customer\_id = orders.customer\_id

INNER JOIN order\_details ON

orders.order\_id = order\_details.order\_id

INNER JOIN products ON

products.product\_id = order\_details.product\_id

WHERE orders.order\_date BETWEEN '1/1/2001' AND '12/31/2001'

GO

--B4

SELECT orders.customer\_id,

customers.name,

customers.phone,

orders.order\_id,

orders.order\_date

FROM orders

INNER JOIN customers ON

orders.customer\_id = customers.customer\_id

WHERE orders.shipped\_date IS NULL

ORDER BY customers.name

GO

--B5

SELECT customers.customer\_id,

customers.name,

customers.city,

titles.description

FROM customers

INNER JOIN titles ON

customers.title\_id = titles.title\_id

WHERE customers.region IS NULL

GO

--B6

SELECT 'supplier\_name' = suppliers.name,

'product\_name' = products.name,

products.reorder\_level,

products.quantity\_in\_stock

FROM products

INNER JOIN suppliers ON

suppliers.supplier\_id = products.supplier\_id

WHERE products.reorder\_level > products.quantity\_in\_stock

ORDER BY supplier\_name

GO

--B7

SELECT orders.order\_id,

customers.name,

customers.contact\_name,

'shipped\_date' = CONVERT(char(12),orders.shipped\_date,109),

'elapsed' = DATEDIFF(YEAR,orders.shipped\_date,'1/1/2008')

FROM orders

INNER JOIN customers ON

customers.customer\_id = orders.customer\_id

WHERE orders.shipped\_date IS NOT NULL

ORDER BY orders.order\_id, elapsed

GO

--B8

SELECT 'name' = LEFT(customers.name, 1),

'total' = Count(LEFT(customers.name, 1))

FROM customers

GROUP BY LEFT(customers.name, 1)

HAVING COUNT(name) > 1 AND LEFT(customers.name, 1) != 'S'

GO

--B9

SELECT orders.order\_id,

order\_details.quantity,

products.product\_id,

products.reorder\_level,

products.supplier\_id

FROM products

INNER JOIN order\_details ON

order\_details.product\_id = products.product\_id

INNER JOIN orders ON

order\_details.order\_id = orders.order\_id

WHERE quantity > 100

ORDER BY orders.order\_id

GO

--B10

SELECT products.product\_id,

products.name,

products.quantity\_per\_unit,

'unit\_price' = cast(products.unit\_price as decimal(10,4))

FROM products

WHERE products.name LIKE '%tofu%' OR products.name LIKE '%chef%'

ORDER BY products.name

GO

--PART C

--C1

CREATE TABLE employee

(

employee\_id int NOT NULL,

last\_name varchar(30) NOT NULL,

first\_name varchar(15) NOT NULL,

address varchar(30),

city varchar(20),

province char(2),

postal\_code varchar(7),

phone varchar(10),

birth\_date DATETIME NOT NULL

)

GO

--C2

ALTER TABLE employee

ADD PRIMARY KEY(employee\_id);

GO

--C3

BULK INSERT employee

FROM 'C:\TextFiles\employee.txt'

WITH ( CODEPAGE=1252,

DATAFILETYPE = 'char',

FIELDTERMINATOR = '\t',

KEEPNULLS,

ROWTERMINATOR = '\n'

)

GO

ALTER TABLE orders

ADD CONSTRAINT fk\_orders\_employee FOREIGN KEY (employee\_id)

REFERENCES employee

(employee\_id);

GO

SELECT \* FROM employee

GO

--C4

INSERT INTO shippers(name)

VALUES('Quick Express');

SELECT \* FROM shippers

GO

--C5

UPDATE products

SET unit\_price = unit\_price \* 1.05

WHERE unit\_price BETWEEN 5 AND 10

GO

SELECT \* FROM products

WHERE unit\_price BETWEEN 5 AND 10.5

GO

--C6

UPDATE customers

SET fax = 'Unknown'

WHERE fax IS NULL

GO

SELECT \* FROM customers

WHERE fax = 'Unknown'

GO

--C7

CREATE VIEW vw\_order\_cost

AS

SELECT orders.order\_id,

orders.order\_date,

products.product\_id,

customers.name,

'order\_cost' = cast(order\_details.quantity \* products.unit\_price as decimal(10,4))

FROM orders

INNER JOIN customers ON

customers.customer\_id = orders.customer\_id

INNER JOIN order\_details ON

order\_details.order\_id = orders.order\_id

INNER JOIN products ON

products.product\_id = order\_details.product\_id

GO

SELECT \* FROM vw\_order\_cost

WHERE order\_id BETWEEN 10000 AND 10200

GO

--C8

CREATE VIEW vw\_list\_employees

AS

SELECT \* FROM employee

GO

SELECT employee\_id,

'name' = last\_name + ', ' + first\_name,

'birth\_date' = CONVERT(char(10),birth\_date,102)

FROM vw\_list\_employees

WHERE employee\_id = 5 OR employee\_id = 7 OR employee\_id = 9

GO

--C9

CREATE VIEW vw\_all\_orders

AS

SELECT orders.order\_id,

customers.customer\_id,

'customer\_name' = customers.name,

customers.city,

customers.country,

orders.shipped\_date

FROM orders

INNER JOIN customers ON

customers.customer\_id = orders.customer\_id

GO

SELECT order\_id,

customer\_id,

customer\_name,

city,

country,

'shipped\_date' = CONVERT(char(11),shipped\_date,100)

FROM vw\_all\_orders

WHERE shipped\_date BETWEEN '1/1/2002' AND '12/31/2002'

ORDER BY customer\_name, country

GO

--C10

CREATE VIEW c10

AS

SELECT suppliers.supplier\_id,

'suppliers\_name' = suppliers.name,

products.product\_id,

'product\_name' = products.name

FROM products

INNER JOIN suppliers ON

suppliers.supplier\_id = products.supplier\_id

GO

SELECT \* FROM c10

GO

--PART D

--D1

CREATE PROCEDURE sp\_customer\_city

(

@city varchar(60)

)

AS

SELECT customer\_id,

name,

address,

city,

phone

FROM customers

WHERE city = @city

GO

EXECUTE sp\_customer\_city 'London'

GO

--D2

CREATE PROCEDURE sp\_orders\_by\_dates

(

@start DATETIME,

@end DATETIME

)

AS

SELECT orders.order\_id,

customers.customer\_id,

'customer\_name' = customers.name,

'shipper\_name' = shippers.name,

orders.shipped\_date

FROM orders

INNER JOIN shippers ON

shippers.shipper\_id = orders.shipper\_id

INNER JOIN customers ON

orders.customer\_id = customers.customer\_id

WHERE orders.shipped\_date BETWEEN @start AND @end

GO

EXECUTE sp\_orders\_by\_dates '1/1/2003', '6/30/2003'

GO

--D3

CREATE PROCEDURE sp\_product\_listing

(

@product varchar(10),

@month varchar(10),

@year int

)

AS

SELECT 'product\_name' = products.name,

products.unit\_price,

products.quantity\_in\_stock,

'supplier\_name' = suppliers.name

FROM products

INNER JOIN suppliers ON

suppliers.supplier\_id = products.supplier\_id

INNER JOIN order\_details ON

order\_details.product\_id = products.product\_id

INNER JOIN orders ON

order\_details.order\_id = orders.order\_id

WHERE products.name LIKE '%' + @product + '%'

AND DATENAME(MONTH, orders.order\_date) = @month

AND DATEPART(YEAR, orders.order\_date) = @year

GO

EXECUTE sp\_product\_listing 'Jack', 'June', '2001'

GO

--D4

CREATE TRIGGER del\_order\_details

ON order\_details

INSTEAD OF DELETE

AS

DECLARE @oid otherid,

@pid otherid

SELECT @oid = order\_id,

@pid = product\_id

FROM DELETED

IF EXISTS

(

SELECT order\_id,

product\_id

FROM order\_details

WHERE order\_id = @oid AND product\_id = @pid

)

BEGIN

SELECT 'Product\_ID' = order\_details.product\_id,

'Product Name' = products.name,

'Quantity being delted from Order' = order\_details.quantity,

'In stock Quantity after Deletion' = products.quantity\_in\_stock + order\_details.quantity

-- NOTE: I'm hesitated to acually delete anything from the database since it might affect

-- the result of other questions

FROM order\_details

INNER JOIN products ON

products.product\_id = order\_details.product\_id

WHERE order\_id = 10001 AND order\_details.product\_id = 25

raiserror('Question D4, not actually deleting',0,1)

ROLLBACK TRANSACTION

END

GO

DELETE order\_details

WHERE order\_id = 10001 AND product\_id = 25

GO

--D5

CREATE TRIGGER tr\_check\_qty

ON order\_details

FOR INSERT, UPDATE

AS

DECLARE @ppid otherid

SELECT @ppid = product\_id

FROM INSERTED

IF

(

SELECT products.units\_on\_order

FROM products

WHERE product\_id = @ppid

)

<=

(

SELECT products.quantity\_in\_stock

FROM products

WHERE products.product\_id = @ppid

)

BEGIN

ROLLBACK TRANSACTION

raiserror('Question D5, Quantity in stock are too low',0,1)

END

GO

UPDATE order\_details

SET quantity = 30

WHERE order\_id = '10044' AND product\_id = 7

GO

--D6

CREATE PROCEDURE sp\_del\_inactive\_cust

AS

DELETE

FROM customers

WHERE customers.customer\_id NOT IN

(

SELECT orders.customer\_id FROM orders

)

GO

EXECUTE sp\_del\_inactive\_cust

GO

--D7

CREATE PROCEDURE sp\_employee\_information

(

@eid int

)

AS

SELECT employee\_id,

last\_name,

first\_name,

address,

city,

province,

postal\_code,

phone,

birth\_date

FROM employee

WHERE @eid = employee\_id

GO

EXECUTE sp\_employee\_information 5

GO

--D8

CREATE PROCEDURE sp\_reorder\_qty

(

@unit int

)

AS

SELECT products.product\_id,

suppliers.name,

suppliers.address,

suppliers.city,

suppliers.province,

'qty' = products.quantity\_in\_stock,

products.reorder\_level

FROM products

INNER JOIN suppliers ON

suppliers.supplier\_id = products.supplier\_id

WHERE (products.quantity\_in\_stock - products.reorder\_level) < @unit

GO

EXECUTE sp\_reorder\_qty 5

GO

--D9

CREATE PROCEDURE sp\_unit\_prices

(

@price1 MONEY,

@price2 MONEY

)

AS

SELECT product\_id,

name,

alternate\_name,

unit\_price

FROM products

WHERE unit\_price BETWEEN @price1 AND @price2

GO

EXECUTE sp\_unit\_prices 5, 10

GO

# 