COMP1630 Project 2

July 7, 2015

By

Bo Liu

Instructor

Mark Bacchus

Table of Contents

1. INTRODUCTIONS

In this project, the author used Microsoft SQL Server to solve the questions from COMP1630. The questions can be separated into 4 parts:

- Part A: Database and Tables
- Part B: SQL Statements
- Part C: INSERT, UPDATE, DELETE and VIEWS Statements
- Part D: Stored Procedures and Triggers

In order to clearly identify the answers to each question, the author stated the question, stated the answer with SQL statement, and showed evidence of the results. The results are captured by screenshots.

All desired results are achieved by the author. One annotated script of all SQL statements can be found at the end.

The following is the table design and the relational diagram of the desired database.

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	No

			Data Type	
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

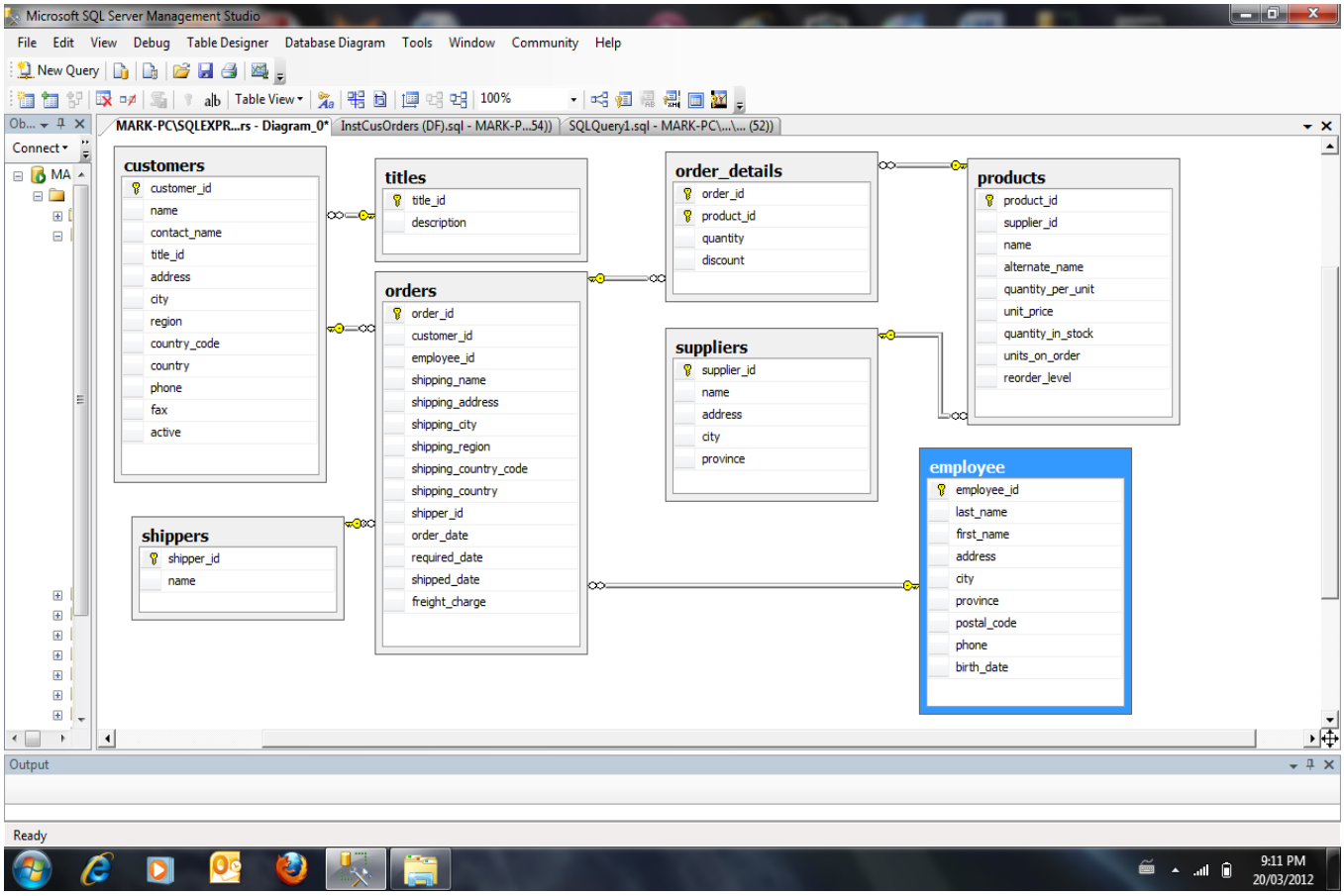
suppliers

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



2. PROJECT SOLUTIONS

2.1. Part A – Database and Tables

2.1.1. Step 1

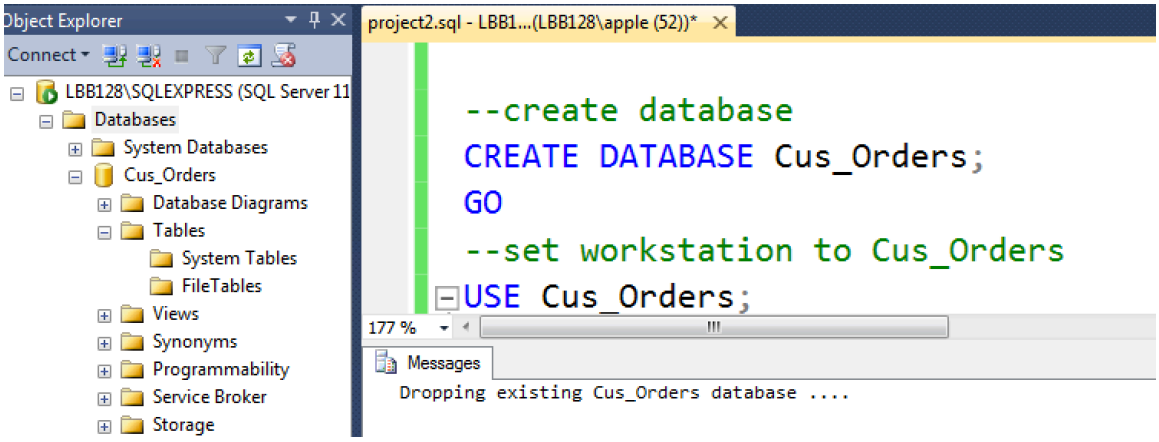
Questions:

Create a database called Cus_Orders

Solutions:

```
/*PART A - DATABASE AND TABLES*/
USE MASTER;
GO

--Drop existing Cus_Orders database
if exists (select * from sysdatabases where name='Cus_Orders')
begin
    raiserror('Dropping existing Cus_Orders database ....',0,1)
    DROP database Cus_Orders
end
GO

--create database
CREATE DATABASE Cus_Orders;
GO
--set workstation to Cus_Orders
USE Cus_Orders;
```

Results:



2.1.2. Step 2

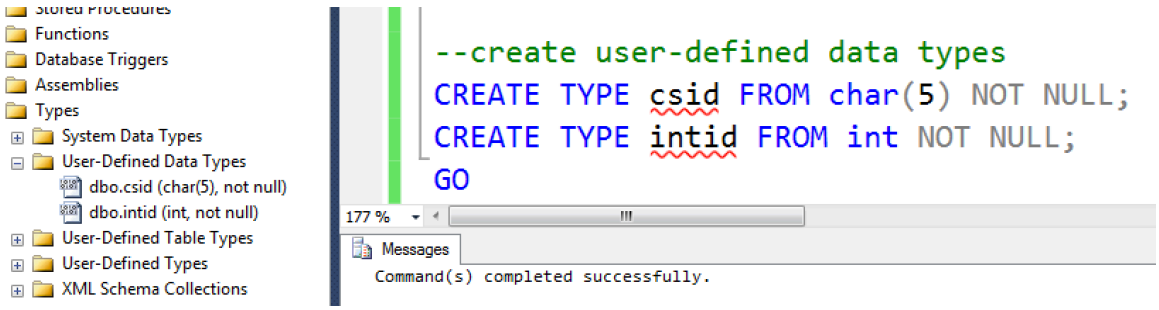
Questions:

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.

Solutions:

```
--create user-defined data types
CREATE TYPE csid FROM char(5) NOT NULL;
CREATE TYPE intid FROM int NOT NULL;
GO
```

Results:



2.1.3. Step 3

Questions:

Create the following tables: customers, orders, order_details, products, shippers, suppliers, titles

Solutions:

```
--create tables
CREATE TABLE customers
(
    customer_id csid,
    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)
);

CREATE TABLE orders
(
    order_id intid,
    customer_id csid,
    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
);

CREATE TABLE order_details
(
    order_id intid,
    product_id intid,
    quantity int NOT NULL,
    discount float NOT NULL
);

CREATE TABLE products
(
    product_id intid,
    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
```

```
);

CREATE TABLE shippers
(
    shipper_id int IDENTITY NOT NULL,
    name varchar(20)
);

CREATE TABLE suppliers
(
    supplier_id int IDENTITY NOT NULL,
    name varchar(40),
    address varchar(30),
    city varchar(20),
    province char(2)
);

CREATE TABLE titles
(
    title_id char(3) NOT NULL,
    description varchar(35)
);
GO
```

2.1.4. Step 4

Questions:

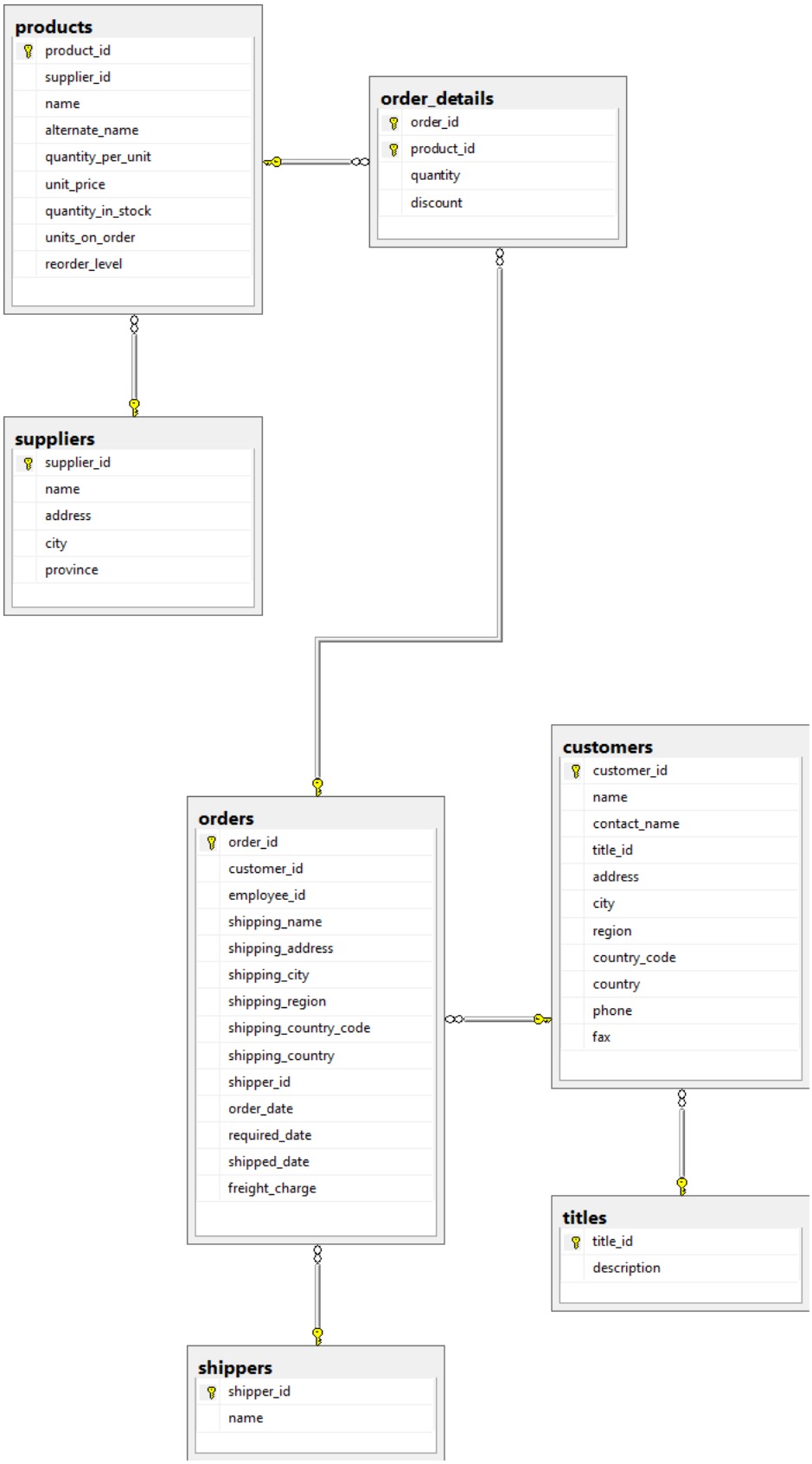
Set the primary keys and foreign keys for the tables.

Solutions:

```
--add primary keys & foreign keys
ALTER TABLE customers
ADD PRIMARY KEY (customer_id)

ALTER TABLE shippers
ADD PRIMARY KEY (shipper_id);

ALTER TABLE titles
ADD PRIMARY KEY (title_id);

ALTER TABLE orders
ADD PRIMARY KEY (order_id);

ALTER TABLE suppliers
ADD PRIMARY KEY (supplier_id);

ALTER TABLE products
ADD PRIMARY KEY (product_id);

ALTER TABLE order_details
ADD PRIMARY KEY (order_id, product_id);
GO

ALTER TABLE customers
ADD CONSTRAINT fk_cus_titles FOREIGN KEY (title_id)
REFERENCES titles(title_id);

ALTER TABLE orders
ADD CONSTRAINT fk_orders_cus FOREIGN KEY (customer_id)
```

```
REFERENCES customers(customer_id);


ALTER TABLE orders
ADD CONSTRAINT fk_orders_shippers FOREIGN KEY (shipper_id)
REFERENCES shippers(shipper_id);


ALTER TABLE order_details
ADD CONSTRAINT fk_orderDetails_orders FOREIGN KEY (order_id)
REFERENCES orders(order_id);


ALTER TABLE order_details
ADD CONSTRAINT fk_orderDetails_products FOREIGN KEY (product_id)
REFERENCES products(product_id);


ALTER TABLE products
ADD CONSTRAINT fk_products_suppliers FOREIGN KEY (supplier_id)
REFERENCES suppliers(supplier_id);
GO
```

Results:


products
product_id
supplier_id
name
alternate_name
quantity_per_unit
unit_price
quantity_in_stock
units_on_order
reorder_level
order_details
order_id
product_id
quantity
discount
suppliers
supplier_id
name
address
city
province
customers
customer_id
name
contact_name
title_id
address
city
region
country_code
country
phone
fax
orders
order_id
customer_id
employee_id
shipping_name
shipping_address
shipping_city
shipping_region
shipping_country_code
shipping_country
shipper_id
order_date
required_date
shipped_date
freight_charge
titles
title_id
description
shippers
shipper_id
name

2.1.5. Step 5

Questions:

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

Solutions:

```
--add constraints to table
ALTER TABLE customers
ADD CONSTRAINT def_country
DEFAULT('Canada') FOR country;

ALTER TABLE orders
ADD CONSTRAINT def_required_date
DEFAULT(GETDATE()+10) FOR required_date;

ALTER TABLE order_details
ADD CONSTRAINT min_quantity
CHECK (quantity>=1);
```

```
ALTER TABLE products
ADD CONSTRAINT min_reorder_level
CHECK (reorder_level >=1);

ALTER TABLE products
ADD CONSTRAINT max_quantity_in_stock
CHECK (quantity_in_stock <150);

ALTER TABLE suppliers
ADD CONSTRAINT def_province
DEFAULT('BC') FOR province;
GO
```

2.1.6. Step 6

Questions:

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	

Solutions:

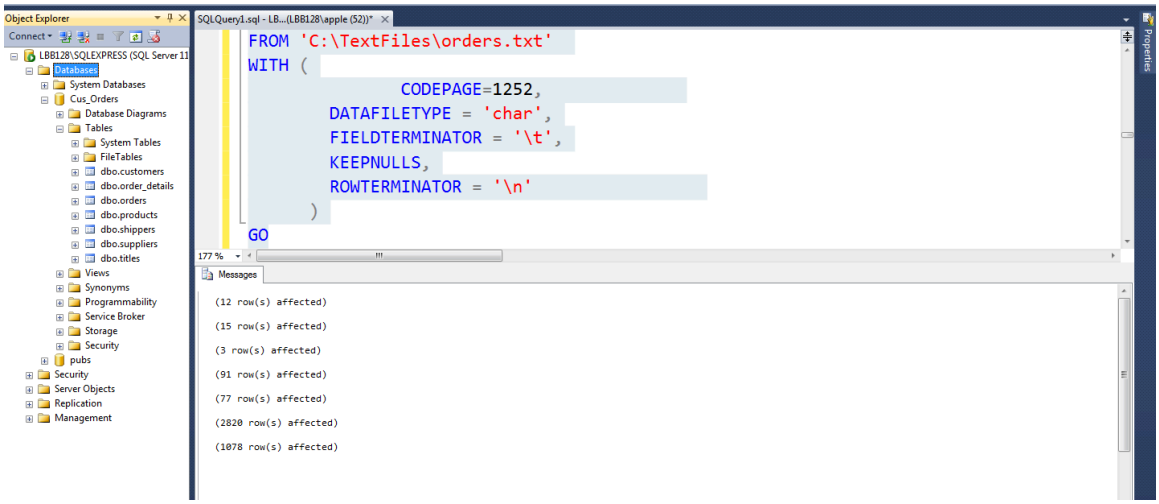
```
--import data from text files
BULK INSERT titles
```

```
FROM 'C:\TextFiles\titles.txt'
WITH (
        CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
     )

BULK INSERT suppliers
FROM 'C:\TextFiles\suppliers.txt'
WITH (
        CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
      )

BULK INSERT shippers
FROM 'C:\TextFiles\shippers.txt'
WITH (
        CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
      )

BULK INSERT customers
```

```
FROM 'C:\TextFiles\customers.txt'
WITH (
        CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
      )

BULK INSERT products
FROM 'C:\TextFiles\products.txt'
WITH (
        CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
      )

BULK INSERT order_details
FROM 'C:\TextFiles\order_details.txt'
WITH (
        CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
      )

BULK INSERT orders
```

```
FROM 'C:\TextFiles\orders.txt'
WITH (
        CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
      )
GO
```

Results:



2.2. Part B – SQL Statements

2.2.1. Step 1

Questions:

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)

Solutions:

```
--part B Step 1

SELECT customer_id, name, city, country

FROM customers

ORDER BY customer_id;
```

Results:

```
--part B Step 1
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
```

77 % | Results | Messages

	customer_id	name	city	country
1	ALFKI	Alfreds Futterkiste	Berlin	Germany
2	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Mexico
3	ANTON	Antonio Moreno Taquería	México D.F.	Mexico
4	AROUT	Around the Horn	London	United Kingdom
5	BERGS	Berglunds snabbköp	Luleå	Sweden
6	BLAUS	Blauer See Delikatessen	Mannheim	Germany
7	BLONP	Blondel père et fils	Strasbourg	France
8	BOLID	Bólido Comidas preparadas	Madrid	Spain
9	BONAP	Bon app'	Marseille	France
10	BOTTM	Bottom-Dollar Markets	Tsawwassen	Canada

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (52) | Cus_Orders | 00:00:00 | 91 rows

```
--part B Step 1
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
```

177 % | Results | Messages

	customer_id	name	city	country
82	TRAIH	Trail's Head Gourmet Provisioners	Kirkland	United States
83	VAFFE	Vaffeljernet	Århus	Denmark
84	VICTE	Victuailles en stock	Lyon	France
85	VINET	Vins et alcools Chevalier	Reims	France
86	WANDK	Die Wandernde Kuh	Stuttgart	Germany
87	WARTH	Wartian Herkku	Oulu	Finland
88	WELLI	Wellington Importadora	Resende	Brazil
89	WHITC	White Clover Markets	Seattle	United States
90	WILMK	Wilman Kala	Helsinki	Finland
91	WOLZA	Wolski Zajazd	Warszawa	Poland

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (52) | Cus_Orders | 00:00:00 | 91 rows

2.2.2. Step 2

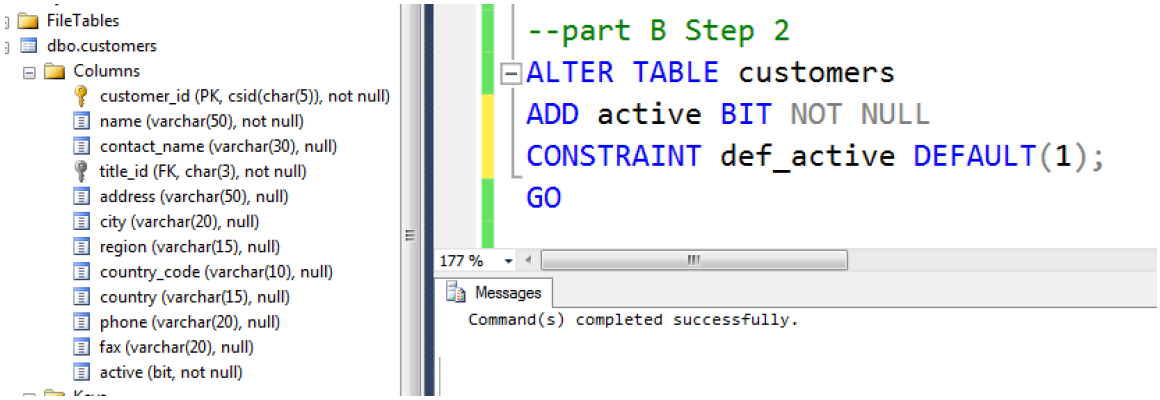
Questions:

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**.

Solutions:

```
--part B Step 2

ALTER TABLE customers

ADD active BIT NOT NULL

CONSTRAINT def_active DEFAULT(1);
```

Results:



2.2.3. Step 3

Questions:

List all the orders where the order date is between **January 1** and **December 31, 2001**. 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

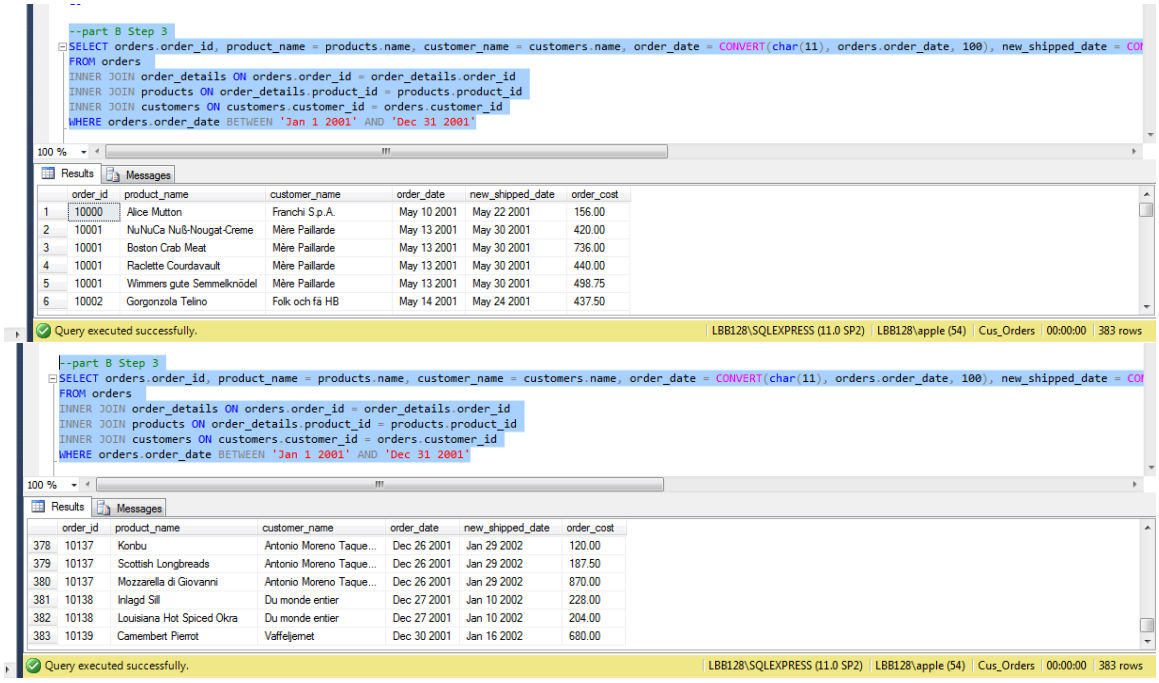
(383 row(s) affected)

Solutions:

```
--part B Step 3
```

```
SELECT orders.order_id, product_name = products.name,
     customer_name = customers.name,
     order_date = CONVERT(char(11), orders.order_date, 100),
     new_shipped_date = CONVERT(char(11),orders.shipped_date + 7,100),
     order_cost = (order_details.quantity * products.unit_price)
FROM orders
INNER JOIN order_details ON orders.order_id = order_details.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON customers.customer_id = orders.customer_id
WHERE orders.order_date BETWEEN 'Jan 1 2001' AND 'Dec 31 2001'
```

Results:



	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
1	10000	Alice Mutton	Franchi S.p.A.	May 10 2001	May 22 2001	156.00
2	10001	NuNuCa Nuß-Nougat-Creme	Mère Paillarde	May 13 2001	May 30 2001	420.00
3	10001	Boston Crab Meat	Mère Paillarde	May 13 2001	May 30 2001	736.00
4	10001	Raclette Courdavault	Mère Paillarde	May 13 2001	May 30 2001	440.00
5	10001	Wimmers gute Semmelknödel	Mère Paillarde	May 13 2001	May 30 2001	498.75
6	10002	Gorgonzola Telino	Folk och fä HB	May 14 2001	May 24 2001	437.50

Query executed successfully. LBB128\SQLEXPRESS (11.0 SP2) LBB128\apple (54) Cus_Orders 00:00:00 383 rows

	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
378	10137	Konbu	Antonio Moreno Taque...	Dec 26 2001	Jan 29 2002	120.00
379	10137	Scottish Longbreads	Antonio Moreno Taque...	Dec 26 2001	Jan 29 2002	187.50
380	10137	Mozzarella di Giovanni	Antonio Moreno Taque...	Dec 26 2001	Jan 29 2002	870.00
381	10138	Inlagd Sill	Du monde entier	Dec 27 2001	Jan 10 2002	228.00
382	10138	Louisiana Hot Spiced Okra	Du monde entier	Dec 27 2001	Jan 10 2002	204.00
383	10139	Camembert Pierrot	Vaffeljernet	Dec 30 2001	Jan 16 2002	680.00

Query executed successfully. LBB128\SQLEXPRESS (11.0 SP2) LBB128\apple (54) Cus_Orders 00:00:00 383 rows

2.2.4. Step 4

Questions:

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. *Your displayed results may look slightly different to those shown below but the query should still return 21 rows*.

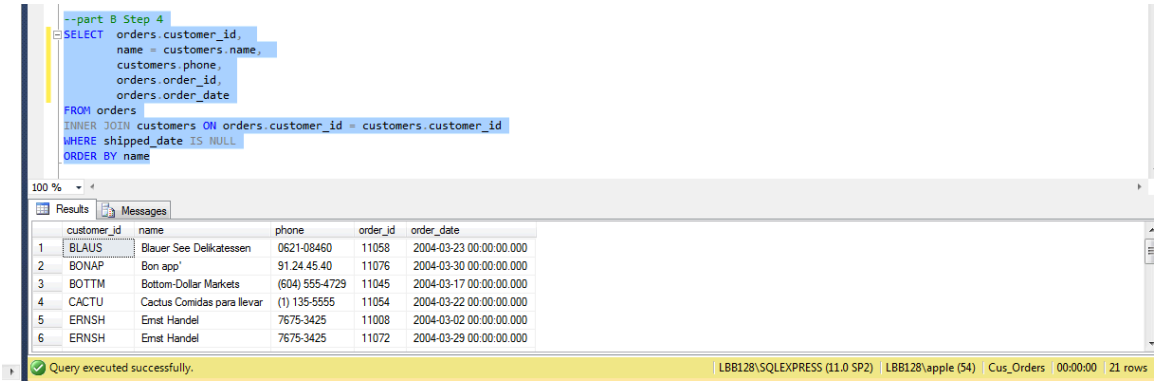
customer_id	name	phone	order_id	order_date
BLAUS	Blauer See Delikatessen	0621-08460	11058	2004-03-23 00:00:00.000
BONAP	Bon app'	91.24.45.40	11076	2004-03-30 00:00:00.000
ERNSH	Ernst Handel	7675-3425	11008	2004-03-02 00:00:00.000
.....				
RICAR	Ricardo Adocicados	(21) 555-3412	11059	2004-03-23 00:00:00.000
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

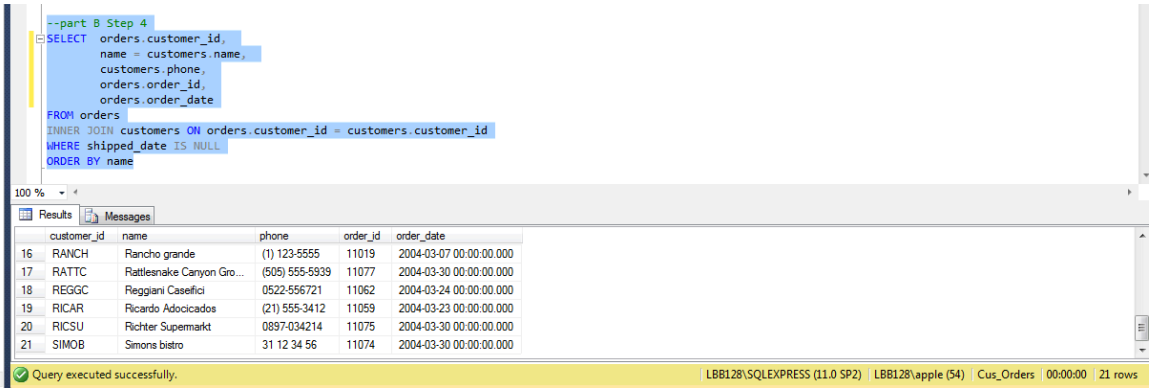
(21 rows affected)

Solutions:

```
SELECT   orders.customer_id,
         name = customers.name,
         customers.phone,
         orders.order_id,
         orders.order_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE shipped_date IS NULL
ORDER BY name
```

Results:





```
--part B Step 4
SELECT  orders.customer_id,
        name = customers.name,
        customers.phone,
        orders.order_id,
        orders.order_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE shipped_date IS NULL
ORDER BY name
```

100 %

Results | Messages

	customer_id	name	phone	order_id	order_date
16	RANCH	Rancho grande	(1) 123-5555	11019	2004-03-07 00:00:00.000
17	RATTC	Rattlesnake Canyon Gro...	(505) 555-5939	11077	2004-03-30 00:00:00.000
18	REGGC	Reggiani Caseifici	0522-556721	11062	2004-03-24 00:00:00.000
19	RICAR	Ricardo Adocicados	(21) 555-3412	11059	2004-03-23 00:00:00.000
20	RICSU	Richter Supermarkt	0897-034214	11075	2004-03-30 00:00:00.000
21	SIMOB	Simons bistro	31 12 34 56	11074	2004-03-30 00:00:00.000

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 21 rows

2.2.5. Step 5

Questions:

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

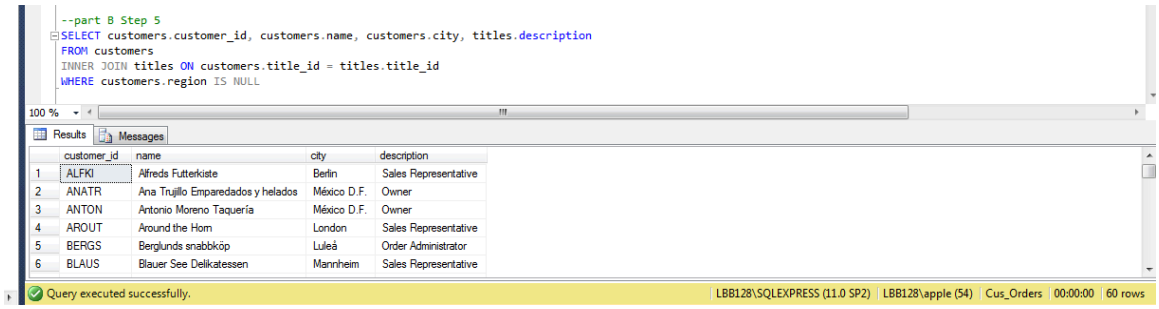
(60 row(s) affected)

Solutions:

```
--part B Step 5

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
```

Results:



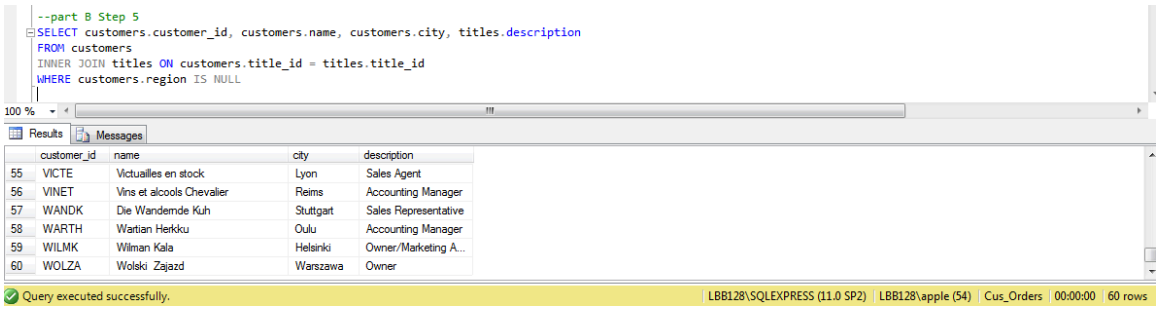
```
--part B Step 5
SELECT customers.customer_id, customers.name, customers.city, titles.description
FROM customers
INNER JOIN titles ON customers.title_id = titles.title_id
WHERE customers.region IS NULL
```

100 %

Results | Messages

	customer_id	name	city	description
1	ALFKI	Alfreds Futterkiste	Berlin	Sales Representative
2	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Owner
3	ANTON	Antonio Moreno Taquería	México D.F.	Owner
4	AROUT	Around the Horn	London	Sales Representative
5	BERGS	Berglunds snabbköp	Luleå	Order Administrator
6	BLAUS	Blauer See Delikatessen	Mannheim	Sales Representative

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 60 rows



```
--part B Step 5
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
```

100 %

Results | Messages

	customer_id	name	city	description
55	VICTE	Victuailles en stock	Lyon	Sales Agent
56	VINET	Vins et alcools Chevalier	Reims	Accounting Manager
57	WANDK	Die Wandernde Kuh	Stuttgart	Sales Representative
58	WARTH	Wartian Herkku	Oulu	Accounting Manager
59	WILMK	Wilman Kala	Helsinki	Owner/Marketing A...
60	WOLZA	Wolski Zajazd	Warszawa	Owner

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 60 rows

2.2.6. Step 6

Questions:

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.

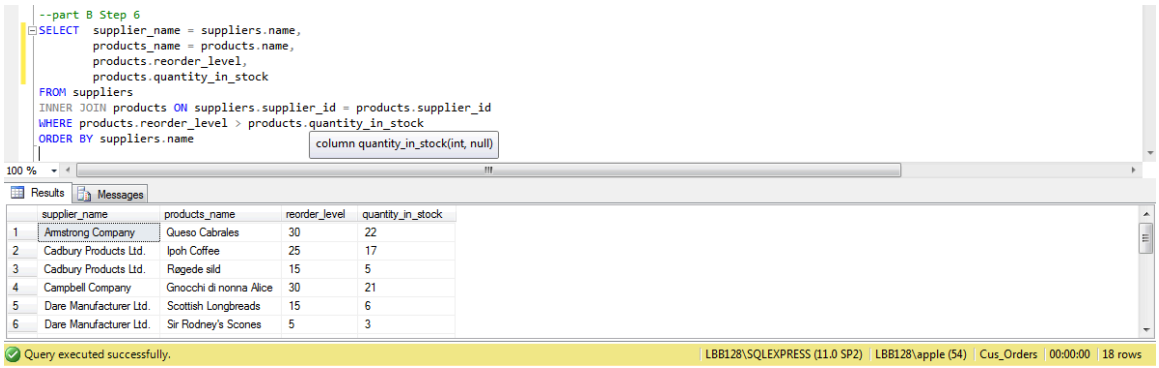
supplier_name	product_name	reorder_level	quantity_in_stock
Armstrong Company	Queso Cabrales	30	22
Cadbury Products Ltd.	Ipoh Coffee	25	17
Cadbury Products Ltd.	Røgede sild	15	5
Campbell Company	Gnocchi di nonna Alice	30	21
Dare Manufacturer Ltd.	Scottish Longbreads	15	6
...			
Steveston Export Company	Gravad lax	25	11
Steveston Export Company	Outback Lager	30	15
Yves Delorme Ltd.	Longlife Tofu	5	4

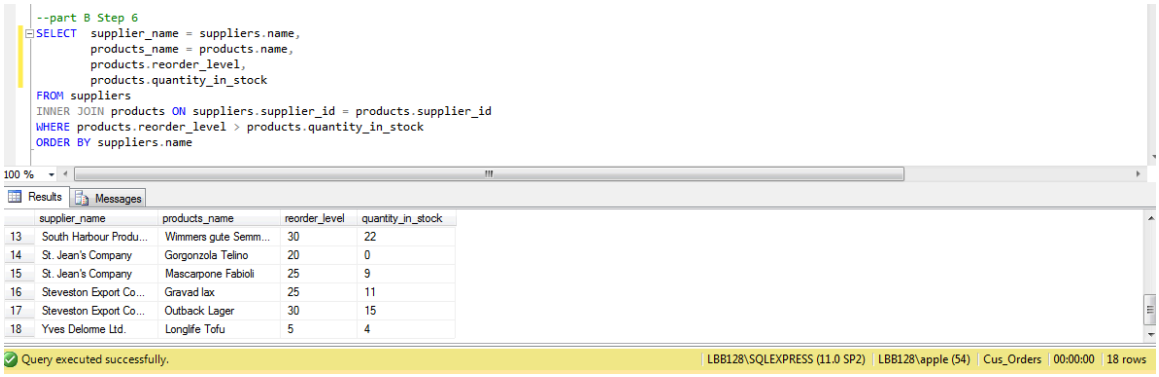
(18 row(s) affected)

Solutions:

```
SELECT   supplier_name = suppliers.name,
         products_name = products.name,
         products.reorder_level,
         products.quantity_in_stock
FROM suppliers
INNER JOIN products ON suppliers.supplier_id = products.supplier_id
WHERE products.reorder_level > products.quantity_in_stock
ORDER BY suppliers.name
```

Results:





```
--part B Step 6
SELECT  supplier_name = suppliers.name,
        products_name = products.name,
        products.reorder_level,
        products.quantity_in_stock
FROM suppliers
INNER JOIN products ON suppliers.supplier_id = products.supplier_id
WHERE products.reorder_level > products.quantity_in_stock
ORDER BY suppliers.name
```

100 %

Results | Messages

	supplier_name	products_name	reorder_level	quantity_in_stock
13	South Harbour Produ...	Wimmers gute Semm...	30	22
14	St. Jean's Company	Gorgonzola Telino	20	0
15	St. Jean's Company	Mascarpone Fabioli	25	9
16	Steveston Export Co...	Gravad lax	25	11
17	Steveston Export Co...	Outback Lager	30	15
18	Yves Delorme Ltd.	Longlife Tofu	5	4

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 18 rows

2.2.7. Step 7

Questions:

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

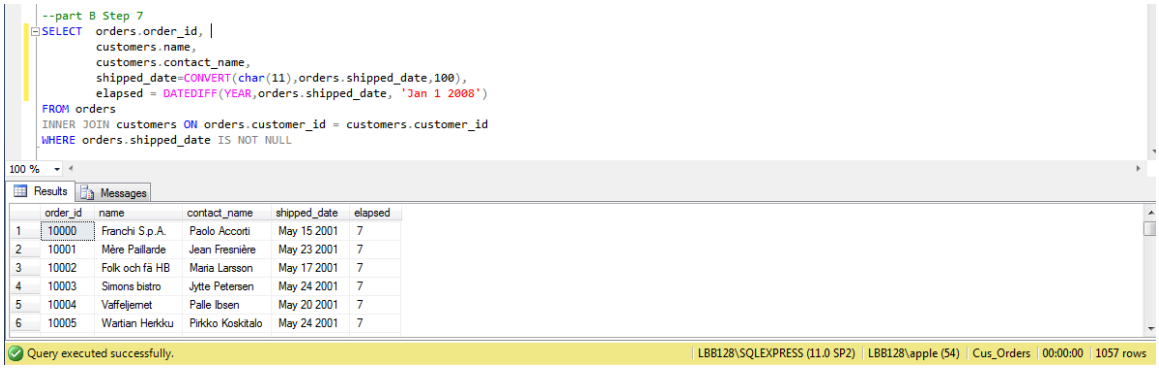
(1057 rows affected)

Solutions:

```
--part B Step 7

SELECT   orders.order_id,

         customers.name,
```

```
        customers.contact_name,
        shipped_date=CONVERT(char(11),orders.shipped_date,100),
        elapsed = DATEDIFF(YEAR,orders.shipped_date, 'Jan 1 2008')
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE orders.shipped_date IS NOT NULL
```

Results:



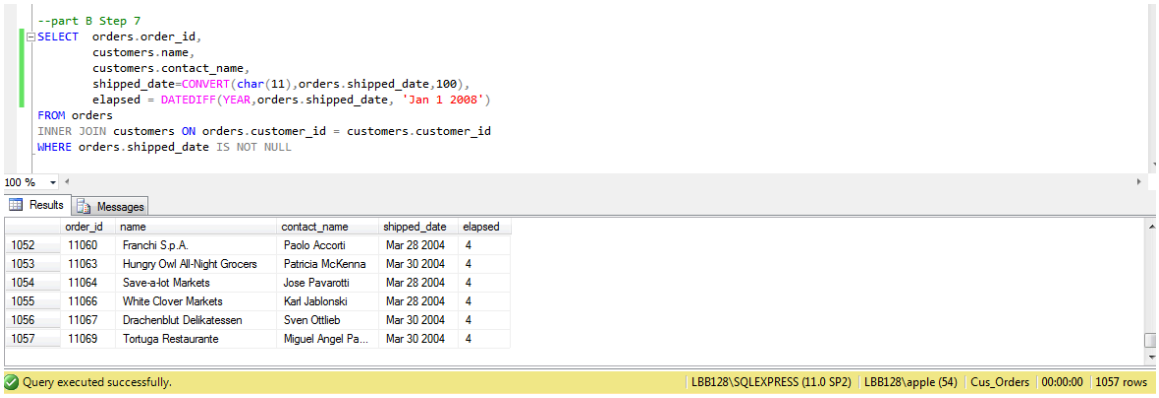
```
--part B Step 7
SELECT  orders.order_id,
        customers.name,
        customers.contact_name,
        shipped_date=CONVERT(char(11),orders.shipped_date,100),
        elapsed = DATEDIFF(YEAR,orders.shipped_date, 'Jan 1 2008')
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE orders.shipped_date IS NOT NULL
```

100 %

Results | Messages

	order_id	name	contact_name	shipped_date	elapsed
1	10000	Franchi S.p.A.	Paolo Accorti	May 15 2001	7
2	10001	Mère Paillarde	Jean Fresnière	May 23 2001	7
3	10002	Folk och fä HB	Maria Larsson	May 17 2001	7
4	10003	Simons bistro	Jytte Petersen	May 24 2001	7
5	10004	Vaffeljernet	Palle Ibsen	May 20 2001	7
6	10005	Wartian Herkku	Pirkko Koskitalo	May 24 2001	7

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 1057 rows



```
--part B Step 7
SELECT  orders.order_id,
        customers.name,
        customers.contact_name,
        shipped_date=CONVERT(char(11),orders.shipped_date,100),
        elapsed = DATEDIFF(YEAR,orders.shipped_date, 'Jan 1 2008')
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE orders.shipped_date IS NOT NULL
```

100 %

Results | Messages

	order_id	name	contact_name	shipped_date	elapsed
1052	11060	Franchi S.p.A.	Paolo Accorti	Mar 28 2004	4
1053	11063	Hungry Owl All-Night Grocers	Patricia McKenna	Mar 30 2004	4
1054	11064	Save-a-lot Markets	Jose Pavarotti	Mar 28 2004	4
1055	11066	White Clover Markets	Karl Jablonski	Mar 28 2004	4
1056	11067	Drachenblut Delikatessen	Sven Ottlieb	Mar 30 2004	4
1057	11069	Tortuga Restaurante	Miguel Angel Pa...	Mar 30 2004	4

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 1057 rows

2.2.8. Step 8

Questions:

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)

Solutions:

```
--part B Step 8

SELECT name = SUBSTRING(name,1,1), 'total' = COUNT(name)

FROM customers

GROUP BY SUBSTRING(name,1,1)

HAVING COUNT(name) >= 2 AND SUBSTRING(name,1,1) != 'S'
```

Results:

2.2.9. Step 9

Questions:

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.

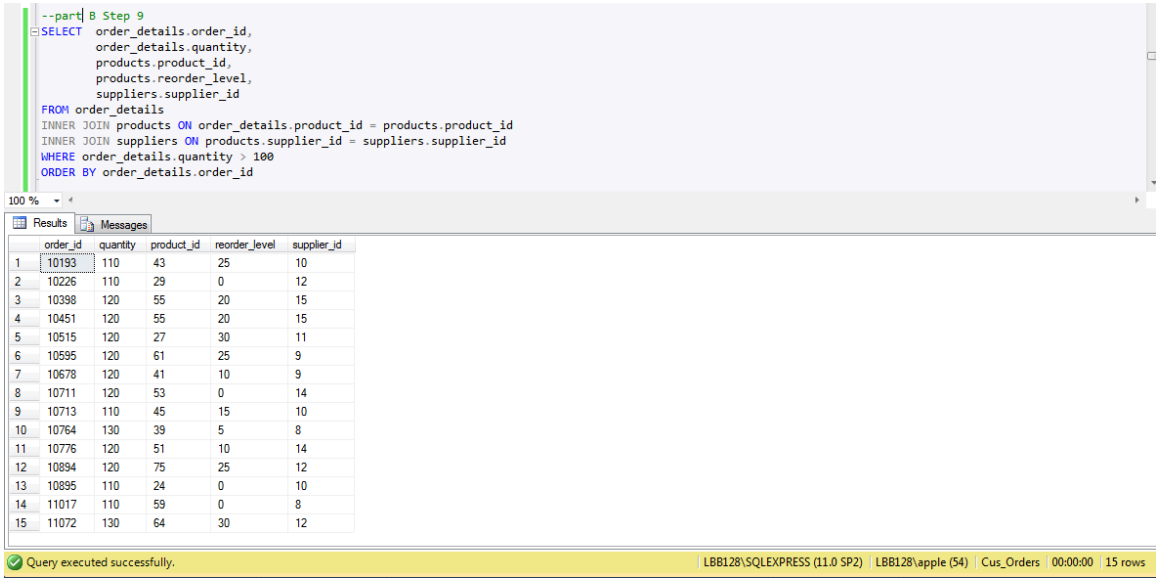
order_id	quantity	product_id	reorder_level	supplier_id
10193	110	43	25	10
10226	110	29	0	12
10398	120	55	20	15
10451	120	55	20	15
10515	120	27	30	11
...				
10895	110	24	0	10
11017	110	59	0	8
11072	130	64	30	12

(15 row(s) affected)

Solutions:

```
--part B Step 9
SELECT   order_details.order_id,
         order_details.quantity,
         products.product_id,
         products.reorder_level,
         suppliers.supplier_id
FROM order_details
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE order_details.quantity > 100
ORDER BY order_details.order_id
```

Results:



```
--part B Step 9
SELECT  order_details.order_id,
        order_details.quantity,
        products.product_id,
        products.reorder_level,
        suppliers.supplier_id
FROM order_details
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE order_details.quantity > 100
ORDER BY order_details.order_id
```

100 %

Results | Messages

	order_id	quantity	product_id	reorder_level	supplier_id
1	10193	110	43	25	10
2	10226	110	29	0	12
3	10398	120	55	20	15
4	10451	120	55	20	15
5	10515	120	27	30	11
6	10595	120	61	25	9
7	10678	120	41	10	9
8	10711	120	53	0	14
9	10713	110	45	15	10
10	10764	130	39	5	8
11	10776	120	51	10	14
12	10894	120	75	25	12
13	10895	110	24	0	10
14	11017	110	59	0	8
15	11072	130	64	30	12

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 15 rows

2.2.10. Step 10

Questions:

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)

Solutions:

```
--part B Step 10

SELECT product_id, name, quantity_per_unit, unit_price
```

```
FROM products
WHERE name LIKE '%tofu%' OR name LIKE '%chef%'
ORDER BY name
```

Results:

2.3. Part C – INSERT, UPDATE, DELETE and VIEWS Statements

2.3.1. Step 1

Questions:

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

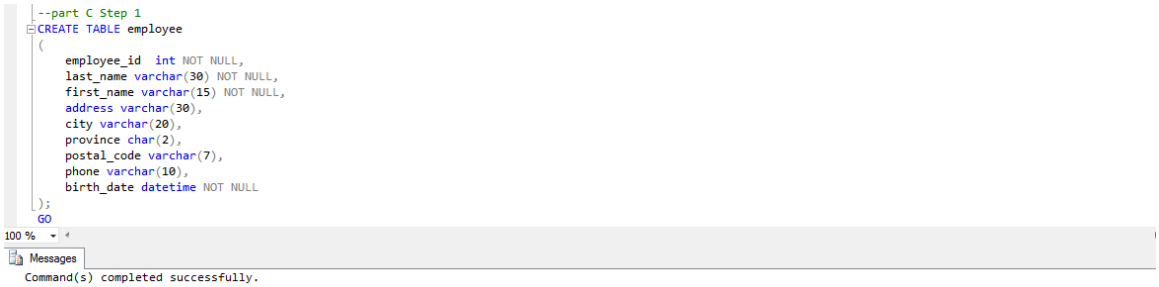
Column Name	*Data Type*	*Length*	*Null Values*
employee_id	int		No
last_name	varchar	30	No
first_name	varchar	15	No
address	varchar	30	
city	varchar	20	
province	char	2	
postal_code	varchar	7	
phone	varchar	10	
birth_date	datetime		No

Solutions:

```
--part C Step 1
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
```

Results:



2.3.2. Step 2

Questions:

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

Solutions:

```
--part C Step 2
ALTER TABLE employee
ADD PRIMARY KEY (employee_id)
GO
```

Results:

Column Name	Data Type	Allow Nulls
employee_id	int	☐
last_name	varchar(30)	☐
first_name	varchar(15)	☐
address	varchar(30)	☑
city	varchar(20)	☑
province	char(2)	☑
postal_code	varchar(7)	☑
phone	varchar(10)	☑
birth_date	datetime	☐
		☐

2.3.3. Step 3

Questions:

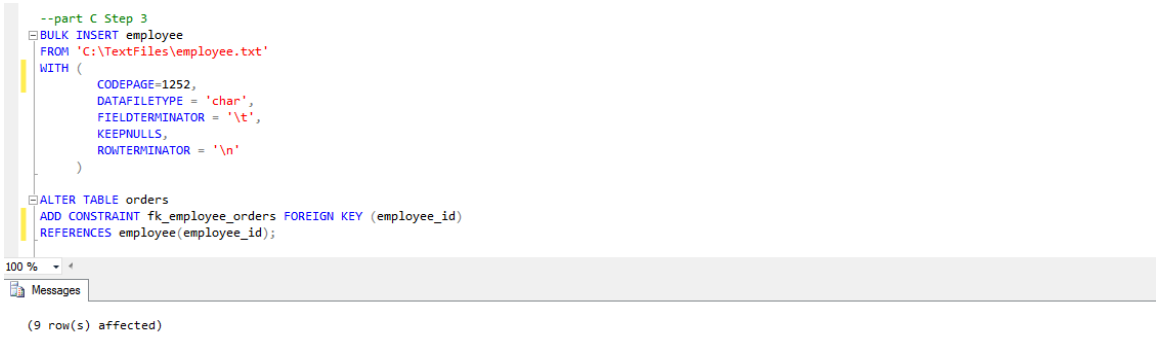
Load the data into the employee table using the employee.txt file; **9** rows. In addition, **create the relationship** to enforce referential integrity between the employee and orders tables.

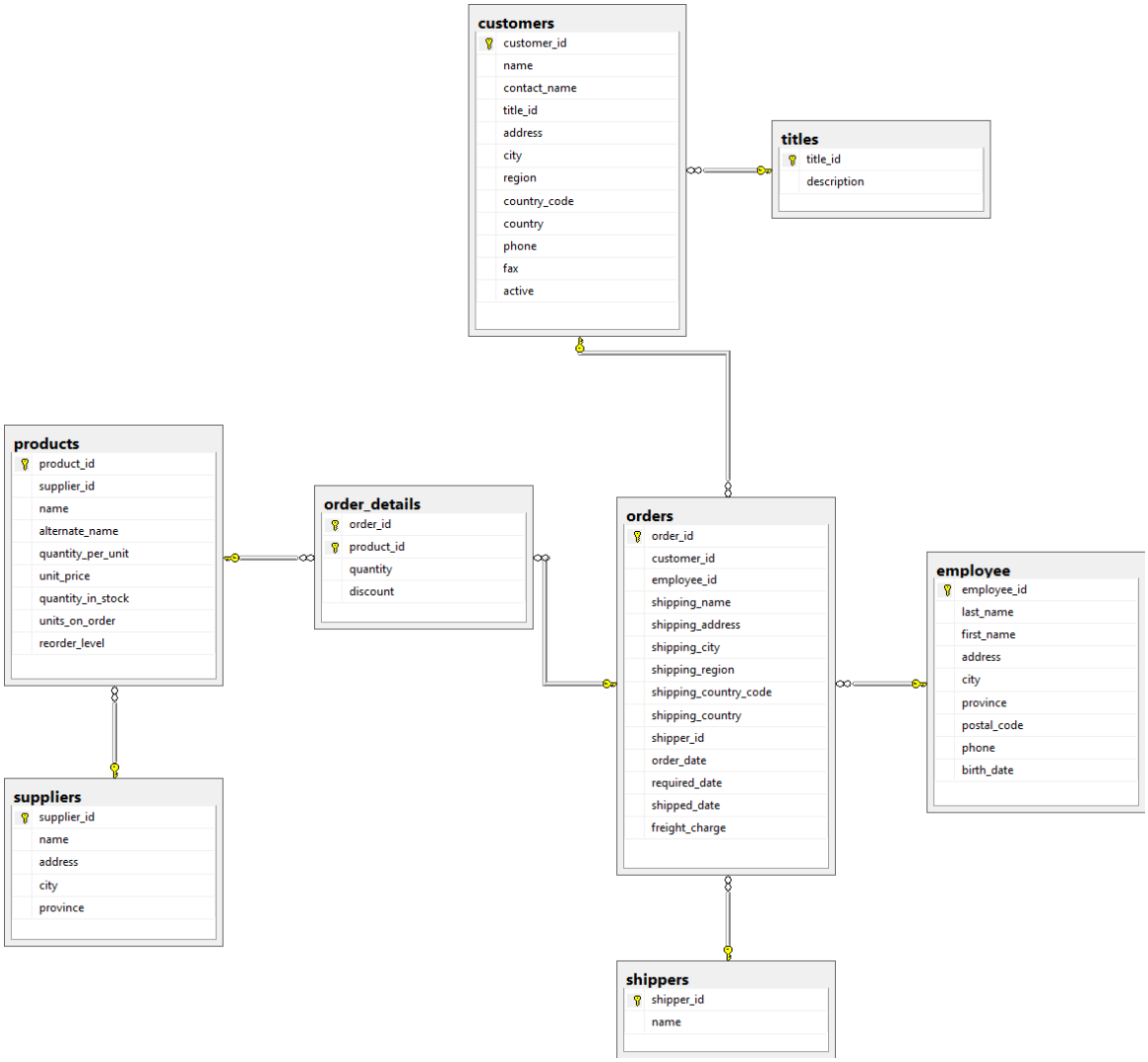
Solutions:

```
--part C Step 3
BULK INSERT employee
FROM 'C:\TextFiles\employee.txt'
WITH (
        CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
 )

ALTER TABLE orders
```

```
ADD CONSTRAINT fk_employee_orders FOREIGN KEY (employee_id)
REFERENCES employee(employee_id);
```

Results:





2.3.4. Step 4

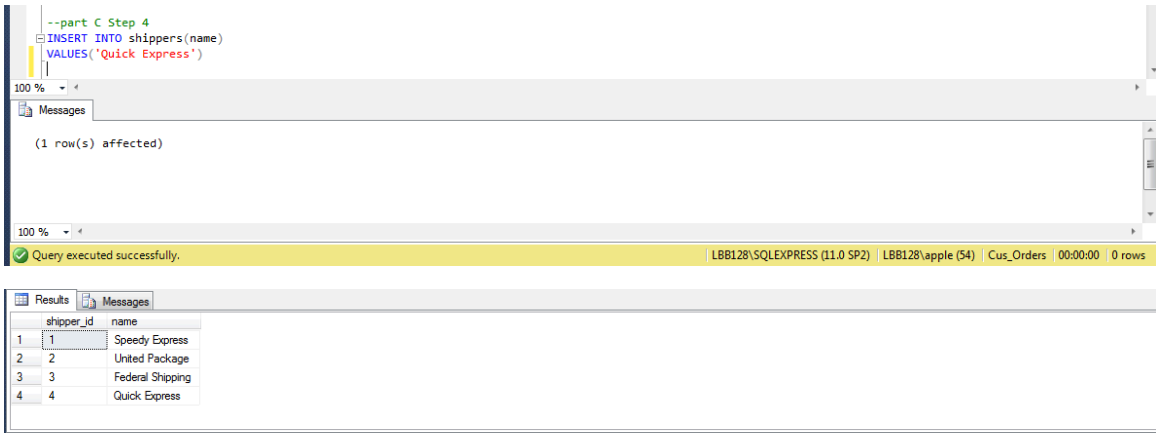
Questions:

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

Solutions:

```
--part C Step 4
INSERT INTO shippers(name)
VALUES('Quick Express')
```

Results:



2.3.5. Step 5

Questions:

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.

Solutions:

```
--part C Step 5

UPDATE products

SET unit_price = unit_price*1.05

WHERE unit_price BETWEEN 5 AND 10
```

Results:

2.3.6. Step 6

Questions:

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.

Solutions:

```
--part C Step 6

UPDATE customers

SET fax = 'Unknown'

WHERE fax IS NULL
```

Results:

2.3.7. Step 7

Questions:

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)

Solutions:

```
--part C Step 7

CREATE VIEW vw_order_cost

AS

SELECT   orders.order_id,

         orders.order_date,

         products.product_id,

         customers.name,

         order_cost=(order_details.quantity*products.unit_price)

FROM orders

INNER JOIN order_details ON order_details.order_id = orders.order_id

INNER JOIN products ON order_details.product_id = products.product_id

INNER JOIN customers ON orders.customer_id = customers.customer_id

GO


SELECT *

FROM vw_order_cost

WHERE order_id BETWEEN 10000 AND 10200
```

Results:

```
--part C Step 7
CREATE VIEW vw_order_cost
AS
SELECT  orders.order_id,
        orders.order_date,
        products.product_id,
        customers.name,
        order_cost=(order_details.quantity*products.unit_price)
FROM orders
INNER JOIN order_details ON order_details.order_id = orders.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO

SELECT *
FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200
GO
```

100 %

Results | Messages

	order_id	order_date	product_id	name	order_cost
1	10000	2001-05-10 00:00:00.000	17	Franchi S.p.A.	156.00
2	10001	2001-05-13 00:00:00.000	25	Mère Paillarde	420.00
3	10001	2001-05-13 00:00:00.000	40	Mère Paillarde	736.00
4	10001	2001-05-13 00:00:00.000	59	Mère Paillarde	440.00
5	10001	2001-05-13 00:00:00.000	64	Mère Paillarde	498.75
6	10002	2001-05-14 00:00:00.000	31	Folk och fä HB	437.50
7	10002	2001-05-14 00:00:00.000	39	Folk och fä HB	324.00

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 540 rows

```
--part C Step 7
CREATE VIEW vw_order_cost
AS
SELECT  orders.order_id,
        orders.order_date,
        products.product_id,
        customers.name,
        order_cost=(order_details.quantity*products.unit_price)
FROM orders
INNER JOIN order_details ON order_details.order_id = orders.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO

SELECT *
FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200
GO
```

100 %

Results | Messages

	order_id	order_date	product_id	name	order_cost
535	10198	2002-03-26 00:00:00.000	56	Océano Atlá...	684.00
536	10198	2002-03-26 00:00:00.000	76	Océano Atlá...	540.00
537	10199	2002-03-27 00:00:00.000	1	Save-a-lot M...	1188.00
538	10199	2002-03-27 00:00:00.000	3	Save-a-lot M...	420.00
539	10199	2002-03-27 00:00:00.000	39	Save-a-lot M...	720.00
540	10200	2002-03-30 00:00:00.000	11	Bólido Comid...	588.00

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 540 rows

2.3.8. Step 8

Questions:

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.

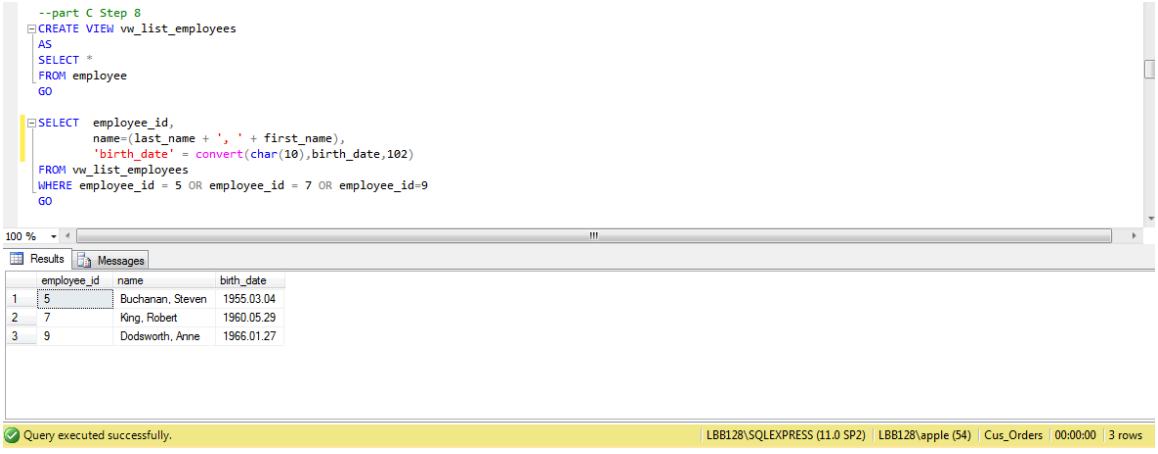
employee_id	name	birth_date
5	Buchanan, Steven	1955.03.04
7	King, Robert	1960.05.29
9	Dodsworth, Anne	1966.01.27

(3 row(s) affected)

Solutions:

```
--part C Step 8

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
```

Results:



2.3.9. Step 9

Questions:

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

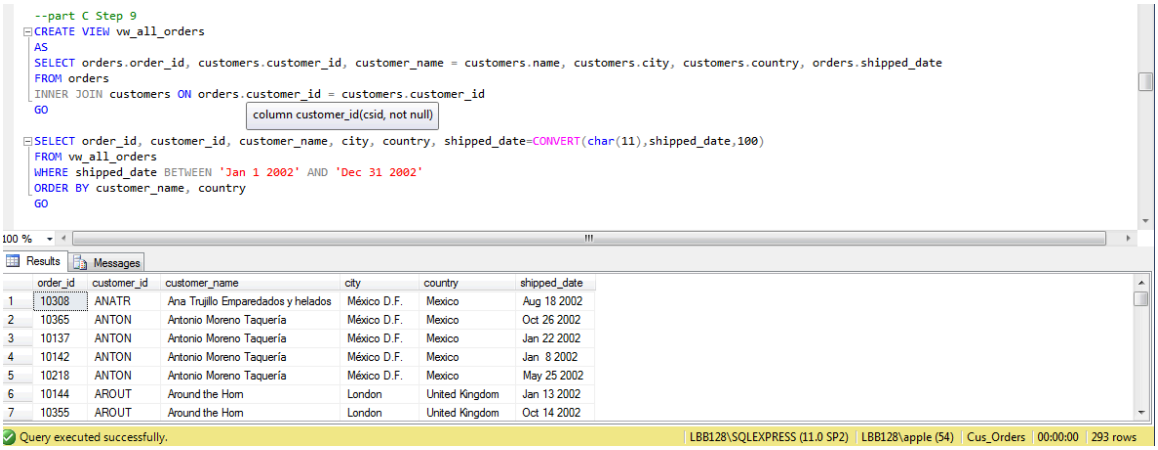
(293 row(s) affected)

Solutions:

```
--part C Step 9

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 orders.customer_id = customers.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 'Jan 1 2002' AND 'Dec 31 2002'
ORDER BY customer_name, country
```

Results:



```
--part C Step 9
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 orders.customer_id = customers.customer_id
GO
```

column customer_id(csid, not null)

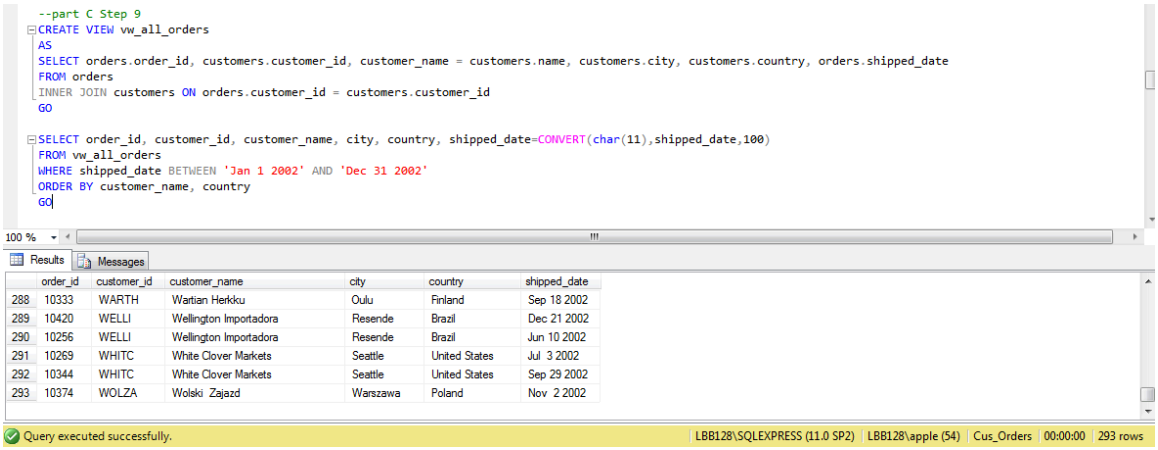
```
SELECT order_id, customer_id, customer_name, city, country, shipped_date=CONVERT(char(11),shipped_date,100)
FROM vw_all_orders
WHERE shipped_date BETWEEN 'Jan 1 2002' AND 'Dec 31 2002'
ORDER BY customer_name, country
GO
```

100 %

Results | Messages

	order_id	customer_id	customer_name	city	country	shipped_date
1	10308	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Mexico	Aug 18 2002
2	10365	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Oct 26 2002
3	10137	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Jan 22 2002
4	10142	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Jan 8 2002
5	10218	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	May 25 2002
6	10144	AROUT	Around the Horn	London	United Kingdom	Jan 13 2002
7	10355	AROUT	Around the Horn	London	United Kingdom	Oct 14 2002

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 293 rows



```
--part C Step 9
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 orders.customer_id = customers.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 'Jan 1 2002' AND 'Dec 31 2002'
ORDER BY customer_name, country
GO
```

100 %

Results | Messages

	order_id	customer_id	customer_name	city	country	shipped_date
288	10333	WARTH	Wartian Herkku	Oulu	Finland	Sep 18 2002
289	10420	WELLI	Wellington Importadora	Resende	Brazil	Dec 21 2002
290	10256	WELLI	Wellington Importadora	Resende	Brazil	Jun 10 2002
291	10269	WHITC	White Clover Markets	Seattle	United States	Jul 3 2002
292	10344	WHITC	White Clover Markets	Seattle	United States	Sep 29 2002
293	10374	WOLZA	Wolski Zajazd	Warszawa	Poland	Nov 2 2002

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 293 rows

2.3.10. Step 10

Questions:

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)

Solutions:

```
--part C Step 10

CREATE VIEW vw_supplier_products

AS

SELECT   suppliers.supplier_id, supplier_name = suppliers.name,

         products.product_id, product_name=products.name

FROM suppliers

INNER JOIN products ON products.supplier_id = suppliers.supplier_id

GO


SELECT *

FROM vw_supplier_products
```

Results:

```
--part C Step 10
CREATE VIEW vw_supplier_products
AS
SELECT suppliers.supplier_id, supplier_name = suppliers.name, products.product_id, product_name=products.name
FROM suppliers
INNER JOIN products ON products.supplier_id = suppliers.supplier_id
GO

SELECT *
FROM vw_supplier_products
GO
```

100 %

Results | Messages

	supplier_id	supplier_name	product_id	product_name
1	1	Edward's Products Ltd.	1	Chai
2	1	Edward's Products Ltd.	2	Chang
3	1	Edward's Products Ltd.	3	Aniseed Syrup
4	2	New Orlean's Spices Ltd.	4	Chef Anton's Cajun Seasoning
5	2	New Orlean's Spices Ltd.	5	Chef Anton's Gumbo Mix
6	3	Macaulay Products Company	6	Grandma's Boysenberry Spread
7	3	Macaulay Products Company	7	Uncle Bob's Organic Dried Pears

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 77 rows

```
--part C Step 10
CREATE VIEW vw_supplier_products
AS
SELECT suppliers.supplier_id, supplier_name = suppliers.name, products.product_id, product_name=products.name
FROM suppliers
INNER JOIN products ON products.supplier_id = suppliers.supplier_id
GO
```

table Cus_Orders.dbo.suppliers

```
SELECT *
FROM vw_supplier_products
GO
```

100 %

Results | Messages

	supplier_id	supplier_name	product_id	product_name
72	14	St. Jean's Company	72	Mozzarella di Giovanni
73	7	Steveston Export Company	73	Röd Kaviar
74	4	Yves Delorme Ltd.	74	Longlife Tofu
75	12	South Harbour Products Ltd.	75	Rhönbräu Klosterbier
76	12	South Harbour Products Ltd.	76	Lakkalikööri
77	12	South Harbour Products Ltd.	77	Original Frankfurter grüne Soße

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 77 rows

2.4. Part D – Stored Procedures and Triggers

2.4.1. Step 1

Questions:

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

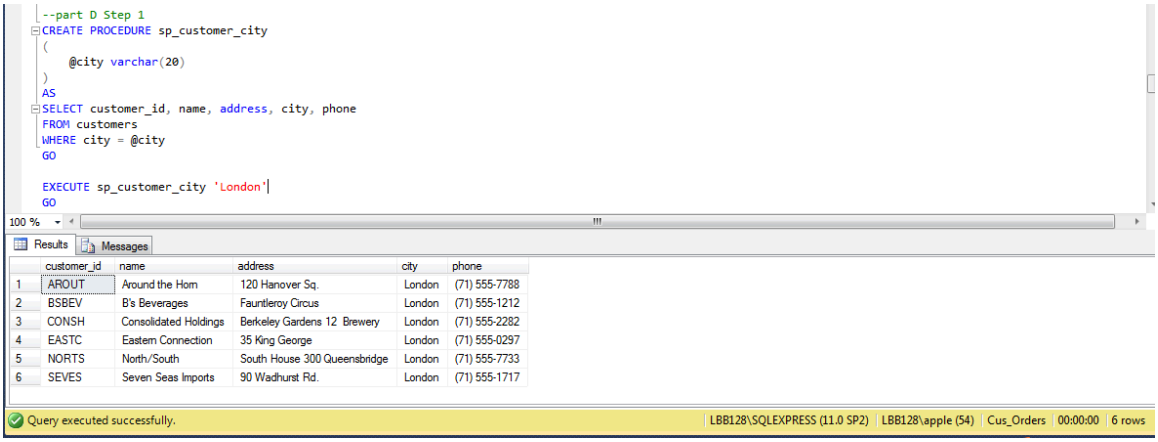
(6 row(s) affected)

Solutions:

```
--part D Step 1

CREATE PROCEDURE sp_customer_city

(

    @city varchar(20)

)

AS

SELECT customer_id, name, address, city, phone

FROM customers

WHERE city = @city

GO
```

```
EXECUTE sp_customer_city 'London'
```

Results:



2.4.2. Step 2

Questions:

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)

Solutions:

```
--part D Step 2

CREATE PROCEDURE sp_orders_by_dates

(

    @start datetime,

    @end datetime

)

AS

SELECT   orders.order_id,

         orders.customer_id,

         customer_name=customers.name,

         shipper_name=shippers.name,

         orders.shipped_date

FROM orders

INNER JOIN customers ON orders.customer_id = customers.customer_id

INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id

WHERE shipped_date BETWEEN @start AND @end

GO


EXECUTE sp_orders_by_dates 'Jan 1 2003', 'Jun 30 2003'
```

Results:

```
--part D Step 2
CREATE PROCEDURE sp_orders_by_dates
(
@start datetime,
@end datetime
)
AS
SELECT  orders.order_id, orders.customer_id,
        customer_name=customers.name, shipper_name=shippers.name, orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id
WHERE shipped_date BETWEEN @start AND @end
GO

EXECUTE sp_orders_by_dates 'Jan 1 2003', 'Jun 30 2003'
GO
```

100 %

Results | Messages

	order_id	customer_id	customer_name	shipper_name	shipped_date
1	10423	GOURL	Gourmet Lanchonetes	Federal Shipping	2003-01-18 00:00:00.000
2	10425	LAMAI	La maison d'Asie	United Package	2003-01-08 00:00:00.000
3	10427	PICCO	Piccolo und mehr	United Package	2003-01-25 00:00:00.000
4	10429	HUNGO	Hungry Owl All-Night Grocers	United Package	2003-01-01 00:00:00.000
5	10431	BOTTM	Bottom-Dollar Markets	United Package	2003-01-01 00:00:00.000
6	10432	SPLIR	Split Rail Beer & Ale	United Package	2003-01-01 00:00:00.000
7	10433	PRINI	Princesa Isabel Vinhos	Federal Shipping	2003-01-26 00:00:00.000

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 188 rows

```
--part D Step 2
CREATE PROCEDURE sp_orders_by_dates
(
@start datetime,
@end datetime
)
AS
SELECT  orders.order_id, orders.customer_id,
        customer_name=customers.name, shipper_name=shippers.name, orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id
WHERE shipped_date BETWEEN @start AND @end
GO

EXECUTE sp_orders_by_dates 'Jan 1 2003', 'Jun 30 2003'
GO
```

100 %

Results | Messages

	order_id	customer_id	customer_name	shipper_name	shipped_date
183	10612	SAVEA	Save-a-lot Markets	United Package	2003-06-25 00:00:00.000
184	10613	HILAA	HILARIÓN-Abastos	United Package	2003-06-25 00:00:00.000
185	10614	BLAUS	Blauer See Delikatessen	Federal Shipping	2003-06-25 00:00:00.000
186	10615	WILMK	Wilman Kala	Federal Shipping	2003-06-30 00:00:00.000
187	10616	GREAL	Great Lakes Food Market	United Package	2003-06-29 00:00:00.000
188	10617	GREAL	Great Lakes Food Market	United Package	2003-06-28 00:00:00.000

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 188 rows

2.4.3. Step 3

Questions:

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) affected)

Solutions:

```
--part D Step 3

CREATE PROCEDURE sp_product_listing

(

    @product varchar(40),

    @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 products.supplier_id = suppliers.supplier_id

INNER JOIN order_details ON products.product_id = order_details.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 DATENAME(Year, orders.order_date) = @year

GO


EXECUTE sp_product_listing 'Jack', June, 2001
```

Results:

```
--part D Step 3
CREATE PROCEDURE sp_product_listing
(
    @product varchar(40),
    @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 products.supplier_id = suppliers.supplier_id
INNER JOIN order_details ON products.product_id = order_details.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 DATENAME(Year, orders.order_date) = @year
GO

EXECUTE sp_product_listing 'Jack', June, 2001
GO
```

100 %

Results | Messages

	product_name	unit_price	quantity_in_stock	supplier_name
1	Jack's New England Clam Chowder	10.1325	85	Silver Spring Wholesale Market
2	Jack's New England Clam Chowder	10.1325	85	Silver Spring Wholesale Market
3	Jack's New England Clam Chowder	10.1325	85	Silver Spring Wholesale Market
4	Jack's New England Clam Chowder	10.1325	85	Silver Spring Wholesale Market

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (54) | Cus_Orders | 00:00:00 | 4 rows

2.4.4. Step 4

Questions:

Create a **DELETE** trigger called **tr_delete_orders** on the orders table to display an error message if an order is deleted that has a value in the order_details table. (*Since Referential Integrity constraints will normally prevent such deletions, this trigger needs to be an Instead of trigger*.) Run the following query to verify your trigger.

DELETE orders

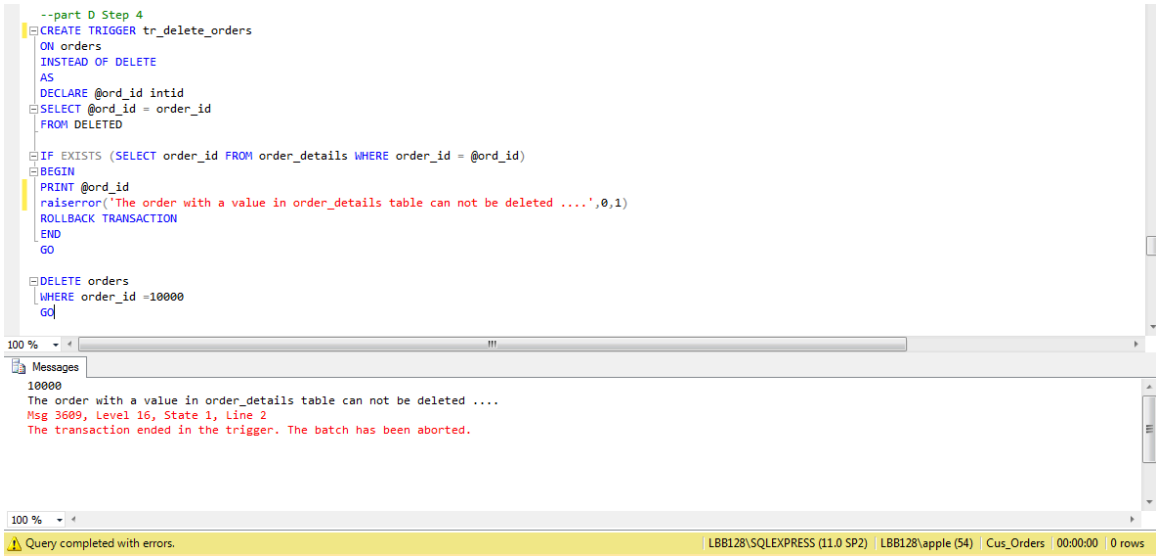
WHERE order_id = 10000

Solutions:

```
--part D Step 4

CREATE TRIGGER tr_delete_orders

ON orders

INSTEAD OF DELETE
```

```
AS
DECLARE @ord_id intid
SELECT @ord_id = order_id
FROM DELETED

IF EXISTS (SELECT order_id FROM order_details WHERE order_id = @ord_id)
BEGIN
raiserror('The order with a value in order_details table can not be
deleted ....',0,1)
ROLLBACK TRANSACTION
END
GO

DELETE orders
WHERE order_id =10000
```

Results:



2.4.5. Step 5

Questions:

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

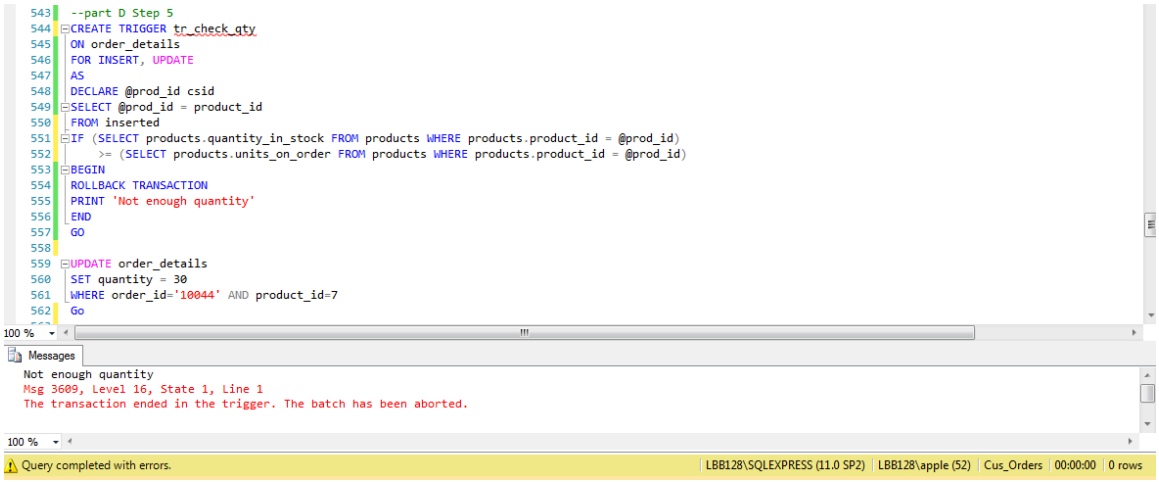
WHERE order_id = '10044' AND product_id = 7

Solutions:

```
--part D Step 5
    CREATE TRIGGER tr_check_qty
    ON order_details
    FOR INSERT, UPDATE
    AS
    DECLARE @prod_id csid
    SELECT @prod_id = product_id
    FROM inserted
    IF (SELECT products.quantity_in_stock FROM products WHERE products.product_id =
    @prod_id)
     >= (SELECT products.units_on_order FROM products WHERE products.product_id =
    @prod_id)
    BEGIN
    ROLLBACK TRANSACTION
    PRINT 'Not enough quantity'
    END
    GO


    UPDATE order_details
    SET quantity = 30
    WHERE order_id='10044' AND product_id=7
```

```
Go
```

Results:



2.4.6. Step 6

Questions:

Create a stored procedure called **sp_del_inactive_cust** to **delete** customers that have no orders. The stored procedure should delete **1** row.

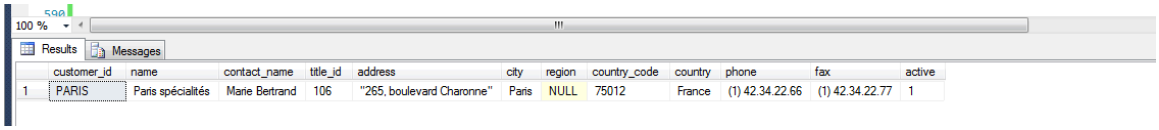
Solutions:

```
--part D Step 6

    CREATE PROCEDURE sp_del_inactive_cust

    AS

    DELETE

    FROM customers

    WHERE customers.customer_id NOT IN

    (

        SELECT orders.customer_id

        FROM orders
```

```
)

EXECUTE sp_del_inactive_cust
GO
```

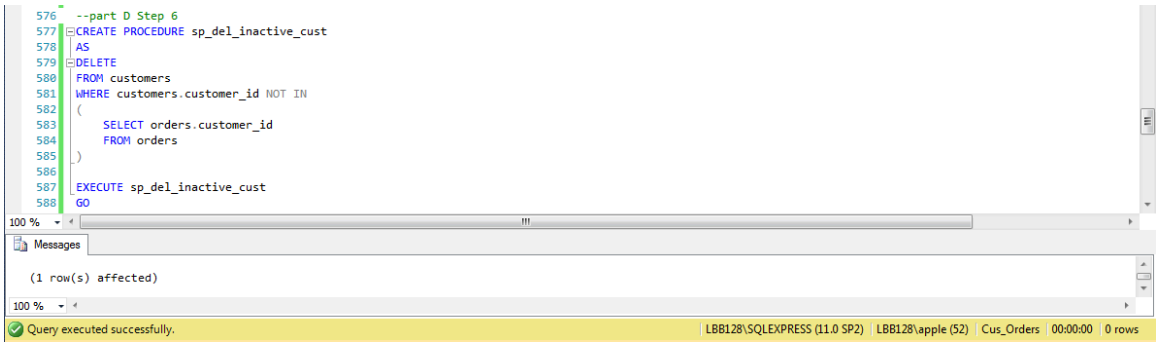
Results:

The customer has no order:



customer_id	name	contact_name	title_id	address	city	region	country_code	country	phone	fax	active
1 PARIS	Paris spécialités	Marie Bertrand	106	"265, boulevard Charonne"	Paris	NULL	75012	France	(1) 42.34.22.66	(1) 42.34.22.77	1

Running the procedure:



```
--part D Step 6
CREATE PROCEDURE sp_del_inactive_cust
 AS
DELETE
 FROM customers
 WHERE customers.customer_id NOT IN
 (
     SELECT orders.customer_id
     FROM orders
 )

 EXECUTE sp_del_inactive_cust
 GO
```

```
(1 row(s) affected)
```

Query executed successfully. LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (52) | Cus_Orders | 00:00:00 | 0 rows

2.4.7. Step 7

Questions:

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.

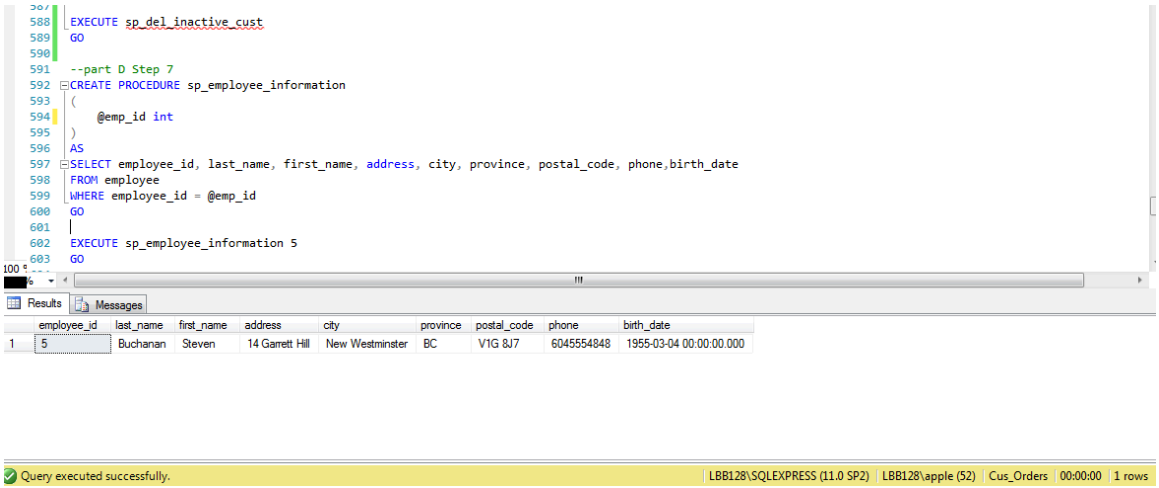
employee_id	last_name	first_name	address	city	province	postal_code	phone	birth_date
5	Buchanan	Steven	14 Garrett Hill	New Westminster	BC	V1G 8J7	6045554848	1955-03-04 00:00:00.000

(1 row(s) affected)

Solutions:

```
--part D Step 7
CREATE PROCEDURE sp_employee_information
(
    @emp_id int
)
AS
SELECT employee_id, last_name, first_name, address, city, province, postal_code,
phone,birth_date
FROM employee
WHERE employee_id = @emp_id
GO

EXECUTE sp_employee_information 5
```

Results:



2.4.8. Step 8

Questions:

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)

Solutions:

```
--part D Step 8

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 products.supplier_id = suppliers.supplier_id

WHERE (products.quantity_in_stock - products.reorder_level) < @unit

GO


EXECUTE sp_reorder_qty 5
```

Results:

```
605  --part D Step 8
606 CREATE PROCEDURE sp_reorder_qty
607  (
608      @unit int
609  )
610  AS
611 SELECT products.product_id, suppliers.name, suppliers.address, suppliers.city, suppliers.province, qty=products.quantity_in_stock, products.reorder_level
612  FROM products
613  INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
614  WHERE (products.quantity_in_stock - products.reorder_level) < @unit
615  GO
616
617  EXECUTE sp_reorder_qty 5
618  GO
```

100 %

Results | Messages

	product_id	name	address	city	province	qty	reorder_level
1	2	Edward's Products Ltd.	1125 Howe Street	Vancouver	BC	17	25
2	3	Edward's Products Ltd.	1125 Howe Street	Vancouver	BC	13	25
3	5	New Orlean's Spices Ltd.	1040 Georgia Street West	Vancouver	BC	0	0
4	11	Armstrong Company	1638 Derwent Way	Richmond	BC	22	30
5	17	Steveston Export Company	2951 Moncton Street	Richmond	BC	0	0
6	21	Dare Manufacturer Ltd.	1603 3rd Avenue West	Burnaby	BC	3	5
7	29	South Harbour Products Ltd.	35 Yale Crescent	Surrey	BC	0	0

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (52) | Cus_Orders | 00:00:00 | 23 rows

```
605  --part D Step 8
606 CREATE PROCEDURE sp_reorder_qty
607  (
608      @unit int
609  )
610  AS
611 SELECT products.product_id, suppliers.name, suppliers.address, suppliers.city, suppliers.province, qty=products.quantity_in_stock, products.reorder_level
612  FROM products
613  INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
614  WHERE (products.quantity_in_stock - products.reorder_level) < @unit
615  GO
616
617  EXECUTE sp_reorder_qty 5
618  GO
```

100 %

Results | Messages

	product_id	name	address	city	province	qty	reorder_level
17	53	St. Jean's Company	119 Cowley Road	Burnaby	BC	0	0
18	56	Campbell Company	892 Farrell Avenue	New We...	BC	21	30
19	64	South Harbour Products Ltd.	35 Yale Crescent	Surrey	BC	22	30
20	66	New Orlean's Spices Ltd.	1040 Georgia Street West	Vancouver	BC	4	20
21	68	Dare Manufacturer Ltd.	1603 3rd Avenue West	Burnaby	BC	6	15
22	70	Steveston Export Company	2951 Moncton Street	Richmond	BC	15	30
23	74	Yves Delorme Ltd.	3050 Granville Street	New We...	BC	4	5

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (52) | Cus_Orders | 00:00:00 | 23 rows

2.4.9. Step 9

Questions:

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

(8 row(s) affected)

Solutions:

```
--part D Step 9

CREATE PROCEDURE sp_unit_prices

(

    @unit1 money,

    @unit2 money

)

AS

SELECT product_id, name, alternate_name, unit_price

FROM products

WHERE unit_price BETWEEN @unit1 AND @unit2

GO


EXECUTE sp_unit_prices 5,10
```

Results:

```
620  --part D Step 9
621 CREATE PROCEDURE sp_unit_prices
622  (
623      @unit1 money,
624      @unit2 money
625  )
626  AS
627 SELECT product_id, name, alternate_name, unit_price
628  FROM products
629  WHERE unit_price BETWEEN @unit1 AND @unit2
630  GO
631
632 EXECUTE sp_unit_prices 5,10
633
```

100 %

Results | Messages

	product_id	name	alternate_name	unit_price
1	13	Konbu	Kelp Seaweed	6.30
2	19	Teatime Chocolate Biscuits	Teatime Chocolate Biscuits	9.66
3	23	Tunnbröd	Thin Bread	9.45
4	45	Røgede sild	Smoked Herring	9.975
5	47	Zaanse koeken	Zaanse Cookies	9.975
6	52	Filo Mix	Mix for Greek Filo Dough	7.35
7	54	Tourtière	Pork Pie	7.8225
8	75	Rhönbräu Klosterbier	Rhönbräu Beer	8.1375

Query executed successfully. | LBB128\SQLEXPRESS (11.0 SP2) | LBB128\apple (52) | Cus_Orders | 00:00:00 | 8 rows

3. SUMMARY

COMP1630 Project 2 is a practical project to solve the relational database questions with all SQL statements learned through the entire class. The author successfully achieves all the desired results and shows the answers with the evidence of results.

4. CHALLENGES

The author spends a lot of time on 2.4.5 and 2.4.6. And also, the errors in the transaction or in the trigger will roll back the entire transactions. The author doesn't use cursor to solve this issue.

5. COPY OF THE SCRIPT

```
/*
    Bo Liu
    A00939260
*/

/*
PART A - DATABASE AND TABLES
*/
USE MASTER;
GO

--Drop existing Cus_Orders database
if exists (select * from sysdatabases where name='Cus_Orders')
begin
    raiserror('Dropping existing Cus_Orders database ....',0,1)
    DROP database Cus_Orders
end
GO

--create database
CREATE DATABASE Cus_Orders;
GO
--set workstation to Cus_Orders
USE Cus_Orders;
GO

--create user-defined data types
```

```
CREATE TYPE csid FROM char(5) NOT NULL;
GO
CREATE TYPE intid FROM int NOT NULL;
GO

--create tables
CREATE TABLE customers
(
    customer_id csid,
    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 intid,
    customer_id csid,
    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 intid,
    product_id intid,
    quantity int NOT NULL,
    discount float NOT NULL
);
GO

CREATE TABLE products
(
    product_id intid,
    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)
);
GO

CREATE TABLE suppliers
(
    supplier_id int IDENTITY NOT NULL,
    name varchar(40),
    address varchar(30),
    city varchar(20),
    province char(2)
);
GO

CREATE TABLE titles
(
    title_id char(3) NOT NULL,
    description varchar(35)
);
GO

--add primary keys & foreign keys
ALTER TABLE customers
```

```
ADD PRIMARY KEY (customer_id)
GO

ALTER TABLE shippers
ADD PRIMARY KEY (shipper_id);
GO

ALTER TABLE titles
ADD PRIMARY KEY (title_id);
GO

ALTER TABLE orders
ADD PRIMARY KEY (order_id);
GO

ALTER TABLE suppliers
ADD PRIMARY KEY (supplier_id);
GO

ALTER TABLE products
ADD PRIMARY KEY (product_id);
GO

ALTER TABLE order_details
ADD PRIMARY KEY (order_id, product_id);
GO

ALTER TABLE customers
ADD CONSTRAINT fk_cus_titles FOREIGN KEY (title_id)
REFERENCES titles(title_id);
```

```
GO

ALTER TABLE orders
ADD CONSTRAINT fk_orders_cus FOREIGN KEY (customer_id)
REFERENCES customers(customer_id);
GO

ALTER TABLE orders
ADD CONSTRAINT fk_orders_shippers FOREIGN KEY (shipper_id)
REFERENCES shippers(shipper_id);
GO

ALTER TABLE order_details
ADD CONSTRAINT fk_orderDetails_orders FOREIGN KEY (order_id)
REFERENCES orders(order_id);
GO

ALTER TABLE order_details
ADD CONSTRAINT fk_orderDetails_products FOREIGN KEY (product_id)
REFERENCES products(product_id);
GO

ALTER TABLE products
ADD CONSTRAINT fk_products_suppliers FOREIGN KEY (supplier_id)
REFERENCES suppliers(supplier_id);
GO

--add constraints to table
ALTER TABLE customers
ADD CONSTRAINT def_country
```

```
DEFAULT('Canada') FOR country;
GO

ALTER TABLE orders
ADD CONSTRAINT def_required_date
DEFAULT(GETDATE()+10) FOR required_date;
GO

ALTER TABLE order_details
ADD CONSTRAINT min_quantity
CHECK (quantity>=1);
GO

ALTER TABLE products
ADD CONSTRAINT min_reorder_level
CHECK (reorder_level >=1);
GO

ALTER TABLE products
ADD CONSTRAINT max_quantity_in_stock
CHECK (quantity_in_stock <150);
GO

ALTER TABLE suppliers
ADD CONSTRAINT def_province
DEFAULT('BC') FOR province;
GO

--import data from text files
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 - SQL STATEMENTS
*/

--part B Step 1
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
GO

--part B Step 2
ALTER TABLE customers
ADD active BIT NOT NULL
```

```
CONSTRAINT def_active DEFAULT(1);
GO

--part B Step 3
SELECT  orders.order_id,
        product_name = products.name,
        customer_name = customers.name,
        order_date = CONVERT(char(11),
        orders.order_date, 100),
        new_shipped_date = CONVERT(char(11),orders.shipped_date + 7,100),
        order_cost = (order_details.quantity * products.unit_price)
FROM orders
INNER JOIN order_details ON orders.order_id = order_details.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON customers.customer_id = orders.customer_id
WHERE orders.order_date BETWEEN 'Jan 1 2001' AND 'Dec 31 2001'
GO

--part B Step 4
SELECT  orders.customer_id,
        name = customers.name,
        customers.phone,
        orders.order_id,
        orders.order_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE shipped_date IS NULL
ORDER BY name
GO
```

```
--part B Step 5
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

--part B Step 6
SELECT  supplier_name = suppliers.name,
        products_name = products.name,
        products.reorder_level,
        products.quantity_in_stock
FROM suppliers
INNER JOIN products ON suppliers.supplier_id = products.supplier_id
WHERE products.reorder_level > products.quantity_in_stock
ORDER BY suppliers.name
GO

--part B Step 7
SELECT  orders.order_id,
        customers.name,
        customers.contact_name,
        shipped_date=CONVERT(char(11),orders.shipped_date,100),
        elapsed = DATEDIFF(YEAR,orders.shipped_date, 'Jan 1 2008')
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE orders.shipped_date IS NOT NULL
```

```
GO


--part B Step 8
SELECT name = SUBSTRING(name,1,1), 'total' = COUNT(name)
FROM customers
GROUP BY SUBSTRING(name,1,1)
HAVING COUNT(name) >= 2 AND SUBSTRING(name,1,1) != 'S'
GO


--part B Step 9
SELECT  order_details.order_id,
        order_details.quantity,
        products.product_id,
        products.reorder_level,
        suppliers.supplier_id
FROM order_details
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE order_details.quantity > 100
ORDER BY order_details.order_id
GO


--part B Step 10
SELECT product_id, name, quantity_per_unit, unit_price
FROM products
WHERE name LIKE '%tofu%' OR name LIKE '%chef%'
ORDER BY name
GO
```

```
/*
PART C - INSERT,UPDATE, DELETE and VIEWS Statements
*/

--part C Step 1
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

--part C Step 2
ALTER TABLE employee
ADD PRIMARY KEY (employee_id)
GO

--part C Step 3
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_employee_orders FOREIGN KEY (employee_id)
REFERENCES employee(employee_id);
GO

--part C Step 4
INSERT INTO shippers(name)
VALUES('Quick Express')
GO

--part C Step 5
UPDATE products
SET unit_price = unit_price*1.05
WHERE unit_price BETWEEN 5 AND 10
GO

--part C Step 6
UPDATE customers
SET fax = 'Unknown'
WHERE fax IS NULL
GO

--part C Step 7
CREATE VIEW vw_order_cost
```

```
AS
SELECT  orders.order_id,
        orders.order_date,
        products.product_id,
        customers.name,
        order_cost=(order_details.quantity*products.unit_price)
FROM orders
INNER JOIN order_details ON order_details.order_id = orders.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON orders.customer_id = customers.customer_id
GO

SELECT *
FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200
GO

--part C Step 8
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
```

```
--part C Step 9
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 orders.customer_id = customers.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 'Jan 1 2002' AND 'Dec 31 2002'
ORDER BY customer_name, country
GO

--part C Step 10
CREATE VIEW vw_supplier_products
AS
SELECT suppliers.supplier_id, supplier_name = suppliers.name, products.product_id,
product_name=products.name
FROM suppliers
INNER JOIN products ON products.supplier_id = suppliers.supplier_id
GO

SELECT *
FROM vw_supplier_products
GO
```

```
/*
PART D - STORED PROCEDURES AND TRIGGERS
*/

--part D Step 1
CREATE PROCEDURE sp_customer_city
(
    @city varchar(20)
)
AS
SELECT customer_id, name, address, city, phone
FROM customers
WHERE city = @city
GO

EXECUTE sp_customer_city 'London'
GO

--part D Step 2
CREATE PROCEDURE sp_orders_by_dates
(
    @start datetime,
    @end datetime
)
AS
SELECT  orders.order_id, orders.customer_id,
        customer_name=customers.name, shipper_name=shippers.name,
orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
```

```
INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id
WHERE shipped_date BETWEEN @start AND @end
GO

EXECUTE sp_orders_by_dates 'Jan 1 2003', 'Jun 30 2003'
GO

--part D Step 3
CREATE PROCEDURE sp_product_listing
(
    @product varchar(40),
    @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 products.supplier_id = suppliers.supplier_id
INNER JOIN order_details ON products.product_id = order_details.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 DATENAME(Year, orders.order_date) = @year
GO

EXECUTE sp_product_listing 'Jack', June, 2001
GO
```

```
--part D Step 4
CREATE TRIGGER tr_delete_orders
ON orders
INSTEAD OF DELETE
AS
DECLARE @ord_id intid
SELECT @ord_id = order_id
FROM DELETED
IF EXISTS (SELECT order_id FROM order_details WHERE order_id = @ord_id)
BEGIN
raiserror('The order with a value in order_details table can not be deleted ....',0,1)
ROLLBACK TRANSACTION
END
GO

DELETE orders
WHERE order_id =10000
GO

--part D Step 5
CREATE TRIGGER tr_check_qty
ON order_details
FOR INSERT, UPDATE
AS
DECLARE @prod_id csid
SELECT @prod_id = product_id
FROM inserted
IF (SELECT products.quantity_in_stock FROM products WHERE products.product_id =
@prod_id)
```

```
    >= (SELECT products.units_on_order FROM products WHERE products.product_id =
@prod_id)
BEGIN
ROLLBACK TRANSACTION
PRINT 'Not enough quantity'
END
GO

UPDATE order_details
SET quantity = 30
WHERE order_id='10044' AND product_id=7
Go

--part D Step 6
CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE
FROM customers
WHERE customers.customer_id NOT IN
(
    SELECT orders.customer_id
    FROM orders
)

EXECUTE sp_del_inactive_cust
GO

--part D Step 7
CREATE PROCEDURE sp_employee_information
(
```

```
    @emp_id int
)
AS
SELECT employee_id, last_name, first_name, address, city, province, postal_code,
phone,birth_date
FROM employee
WHERE employee_id = @emp_id
GO

EXECUTE sp_employee_information 5
GO

--part D Step 8
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 products.supplier_id = suppliers.supplier_id
WHERE (products.quantity_in_stock - products.reorder_level) < @unit
GO

EXECUTE sp_reorder_qty 5
GO

--part D Step 9
CREATE PROCEDURE sp_unit_prices
```

```
(
    @unit1 money,
    @unit2 money
)
AS
SELECT product_id, name, alternate_name, unit_price
FROM products
WHERE unit_price BETWEEN @unit1 AND @unit2
GO

EXECUTE sp_unit_prices 5,10
GO

select * from order_details
GO
```