PROJECT

COMP 1630

By: Duran Sakalli

Instructor: Mark Bacchus

Table of Contents

1. **INTRODUCTION** 4
2. **SOLUTIONS**

Part A – Database and Tables

Step 1 5
Step 2 6
Step 3 7
Step 4 9
Step 5 12
Step 6 14

Part B – SQL Statements

Step 1 17
Step 2 19
Step 3 20
Step 4 22
Step 5 24
Step 6 26
Step 7 28
Step 8 30
Step 9 32
Step 10 34

Part C – INSERT, UPDATE, DELETE and VIEWS Statements

Step 1 35
Step 2 36
Step 3 37
Step 4 38
Step 5 39
Step 6 40
Step 7 40
Step 8 43
Step 9 44
Step 10 46

Part D – Stored Procedures and Triggers

Step 1 ..... **48**

Step 2 ..... **49**

Step 3 ..... **51**

Step 4 ..... **52**

Step 5 ..... **54**

Step 6 ..... **55**

Step 7 ..... **56**

Step 8 ..... **57**

Step 9 ..... **58**

Step 10 ..... **59**

3. CONCLUSION **61**

4. SQL SCRIPTS **62**

1. INTRODUCTION

This database project has been completed by using Microsoft SQL Server Management Studio to create and query a new database. Each question is listed and answered. I have written to execute what has been asked, and finally a snipped showing the successful results.

All steps have been separated into four sections:

- Part A – Database and Tables
  - 6 Steps
- Part B – SQL Statements
  - 10 Steps
- Part C – INSERT, UPDATE, DELETE and VIEWS Statements
  - 10 Steps
- Part D – Stored Procedures and Triggers
  - 9 Steps

At the end, I have included a complete copy of the script of solutions for all steps.

2. SOLUTIONS

Part A – Database and Tables

Step 1

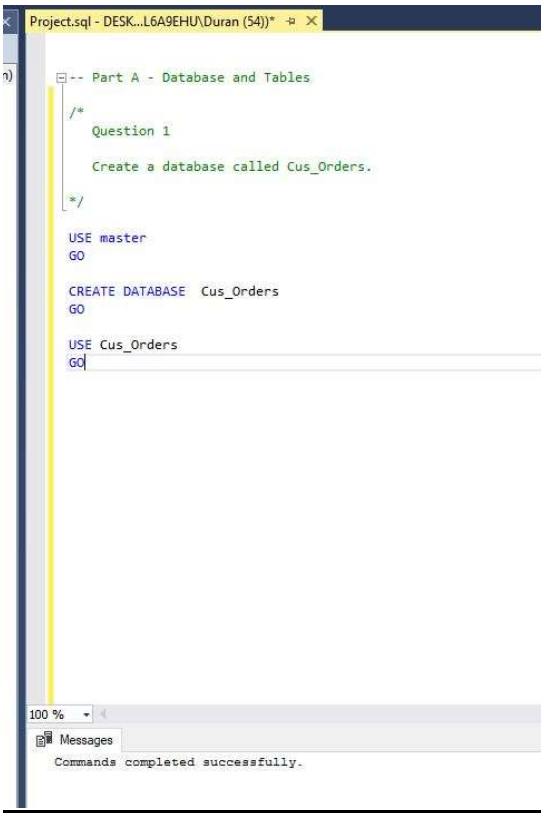
Question:

Create a database called **Cus_Orders**.

Solution:

```
USE master
GO

CREATE DATABASE Cus_Orders
GO

USE Cus_Orders
GO
```

Results:



Step 2

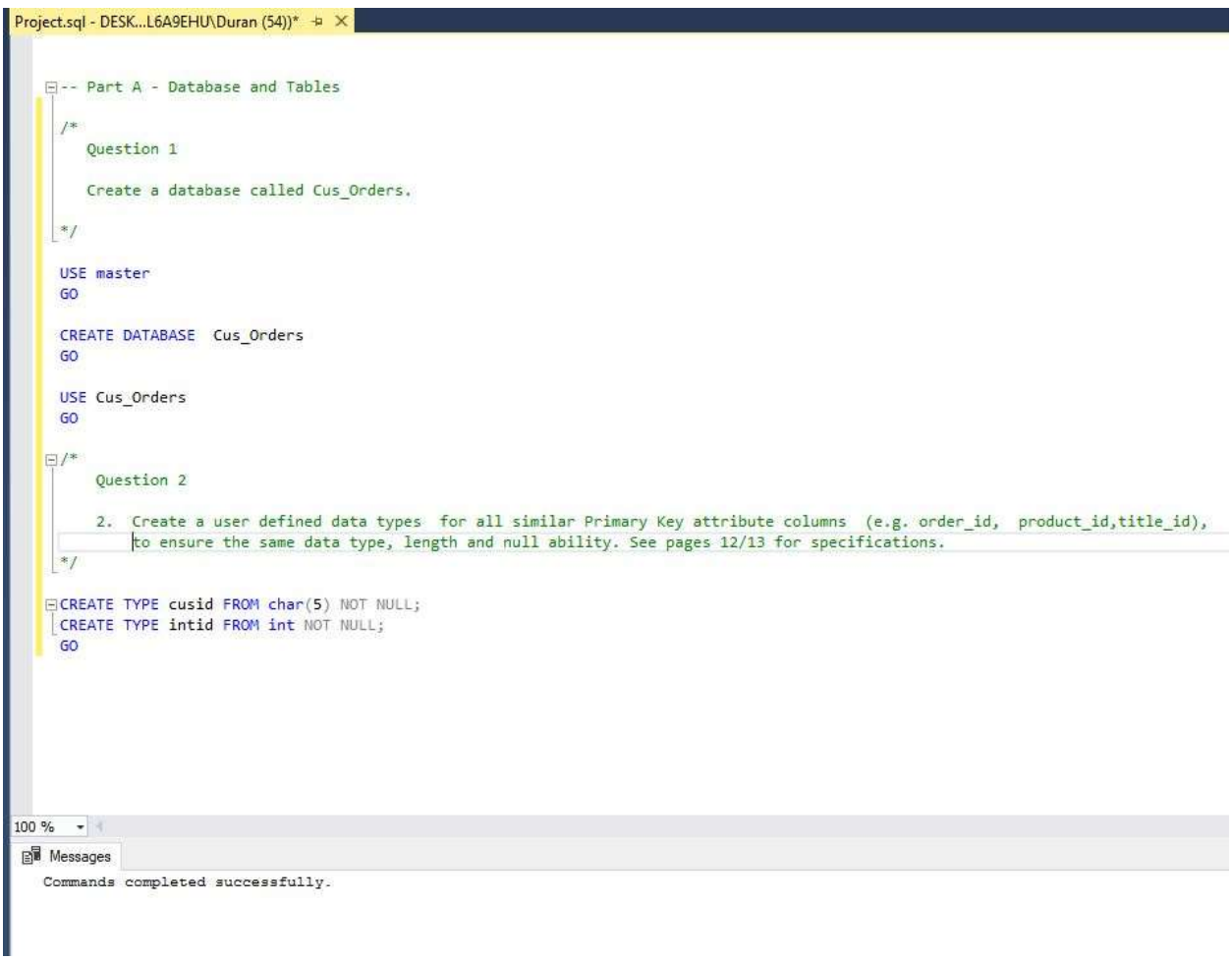
Question:

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.

Solution:

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

Results:



Step 3

Question:

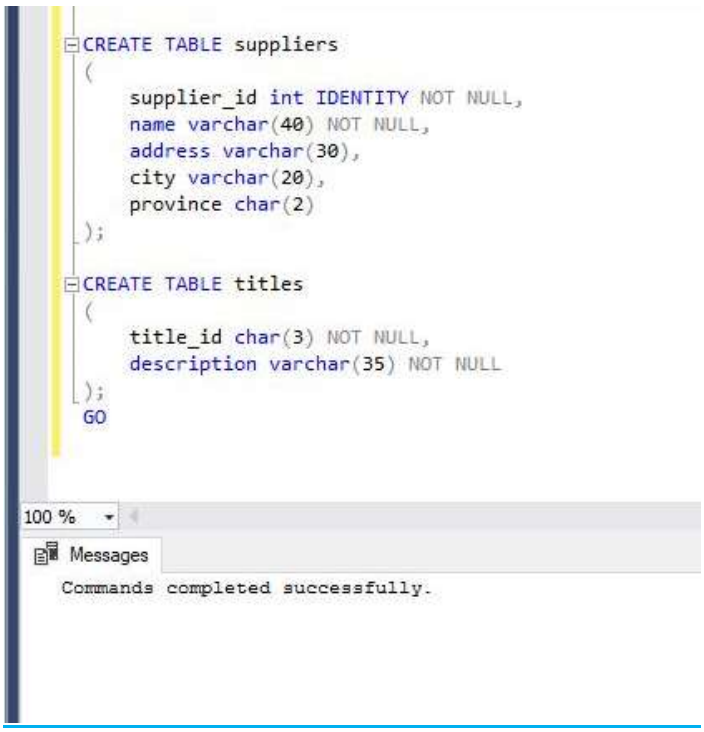
Create the following tables (customers, orders, order_details, products, shippers, suppliers, titles).

Solution:

```
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)
);

CREATE TABLE orders
(
	order_id intid,
	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
);

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) NOT NULL
);

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

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

Results:



Step 4

Question:

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

Solution:

```
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_cust_titles FOREIGN KEY (title_id)
REFERENCES titles(title_id);

ALTER TABLE orders
ADD CONSTRAINT fk_orders_cust 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_order_details_orders FOREIGN KEY (order_id)
REFERENCES orders(order_id);

ALTER TABLE order_details
ADD CONSTRAINT fk_order_details_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:

Project.sql - DESK...L6A9EHU\Duran (54))*

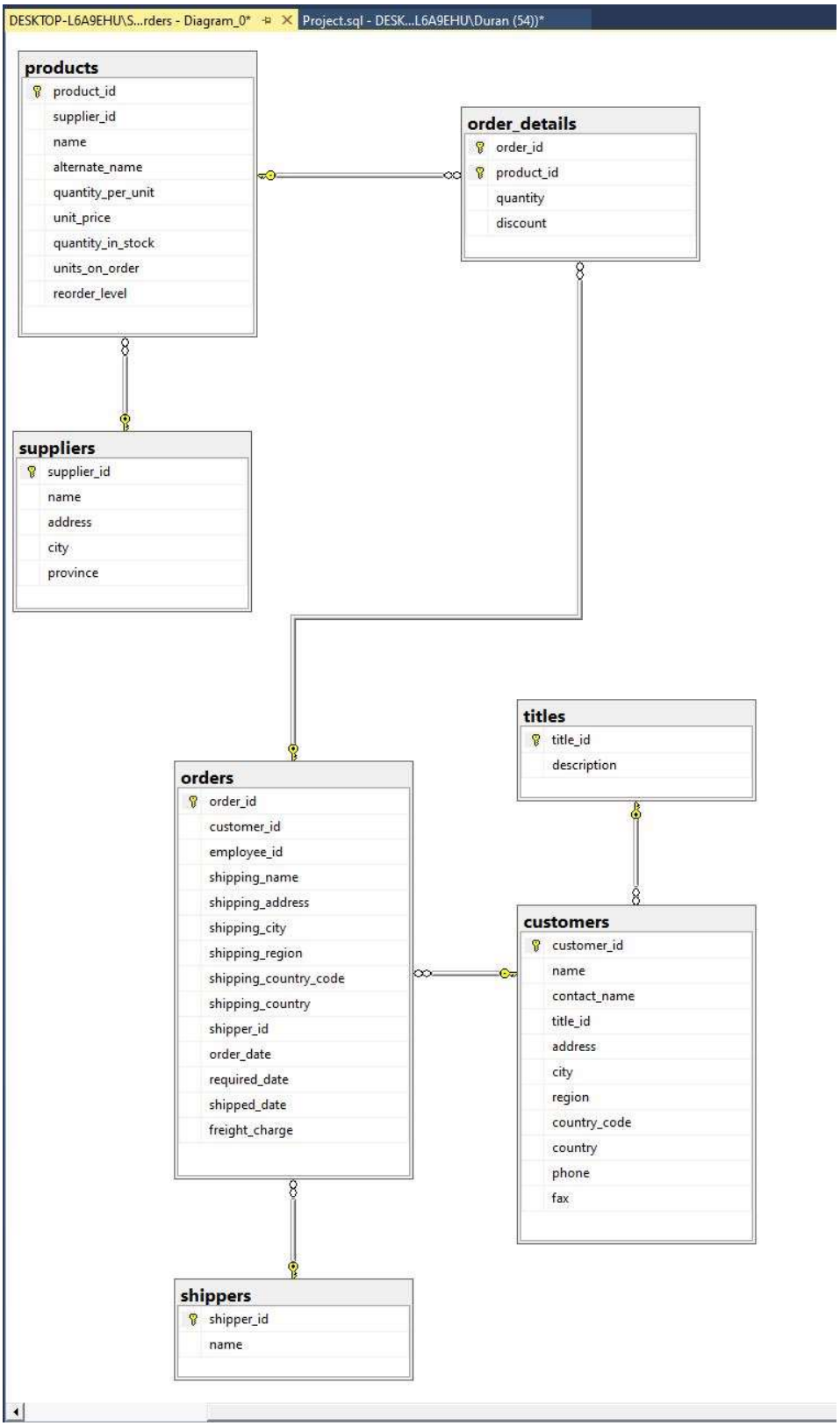
```sql
ADD PRIMARY KEY (supplier_id);

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_cust_titles FOREIGN KEY (title_id)
REFERENCES titles(title_id);

ALTER TABLE orders
ADD CONSTRAINT fk_orders_cust 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_order_details_orders FOREIGN KEY (order_id)
REFERENCES orders(order_id);

ALTER TABLE order_details
ADD CONSTRAINT fk_order_details_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
```

100 %

Messages

```
Commands completed successfully.
```

Database Diagram:



Step 5

Question:

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

Solution:

```
ALTER TABLE customers
ADD CONSTRAINT default_country DEFAULT('Canada') FOR country;

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

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

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

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

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

GO
```

Results:

DESKTOP-L6A9EHU\S...rders - Diagram_0* | Project.sql - DESK...L6A9EHU\Duran (54))*

```
/*
    Question 5

        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
*/

ALTER TABLE customers
ADD CONSTRAINT default_country DEFAULT('Canada') FOR country;

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

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

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

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

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

100 %

Messages

```
Commands completed successfully.
```

Step 6

Question:

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)

Solution:

```
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:

DESKTOP-L6A9EHU\S...rders - Diagram_0* | Project.sql - DESK...L6A9EHU\Duran (54))*

```
        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'
      )
```

100 %

Messages

```
(12 rows affected)

(15 rows affected)

(3 rows affected)

(91 rows affected)

(77 rows affected)

(2820 rows affected)

(1078 rows affected)
```

Part B – SQL Statements

Step 1

Question:

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)

Solution:

```
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
GO
```

Results:

```
--      Part B - SQL Statements

/*
    Question 1
    List the customer id, name, city, and country from the customer table.
    Order the result set by the customer id.
*/

SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
GO

/*
```

100 %

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
11	BSBEV	B's Beverages	London	United Kingdom
12	CACTU	Cactus Comidas para llevar	Buenos Aires	Argentina
13	CENTC	Centro comercial Moctezuma	México D.F.	Mexico
14	CHOPS	Chop-suey Chinese	Bern	Switzerland
15	COMMI	Comércio Mineiro	São Paulo	Brazil
16	CONSH	Consolidated Holdings	London	United Kingdom
17	DRACD	Drachenblut Delikatessen	Aachen	Germany
18	DUMON	Du monde entier	Nantes	France
19	EASTC	Eastern Connection	London	United Kingdom
20	ERNSH	Ernst Handel	Graz	Austria
21	FAMIA	Familia Arquibaldo	São Paulo	Brazil
22	FISSA	FISSA Fabrica Inter. Salchichas S.A.	Madrid	Spain
23	FOLIG	Folies gourmandes	Lille	France
24	FOLKO	Folk och fä HB	Bräcke	Sweden
25	FRANK	Frankenversand	München	Germany

DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 91 rows

Step 2

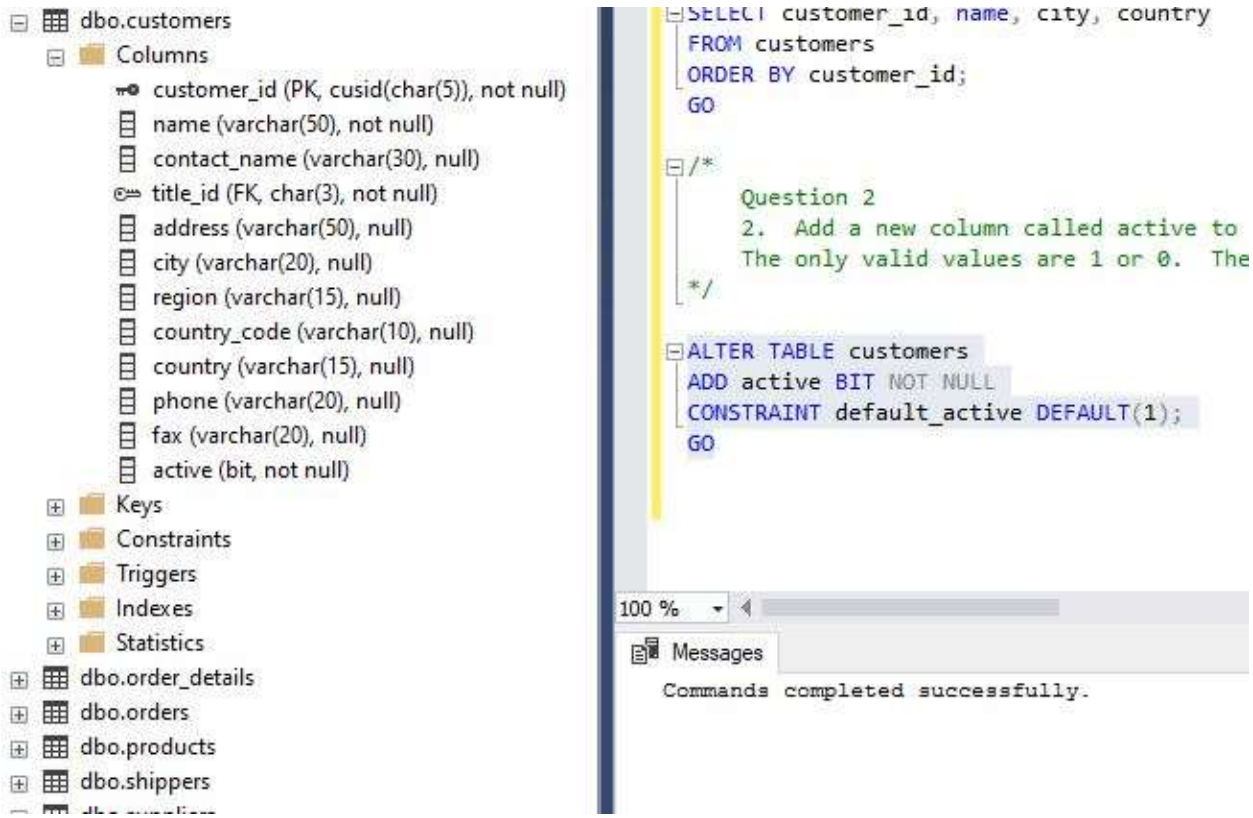
Question:

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**.

Solution:

```
ALTER TABLE customers
ADD active BIT NOT NULL
CONSTRAINT default_active DEFAULT(1);
GO
```

Results:



Step 3

Question:

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 17 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
1	10000	Alice Mutton	Franchi S.p.A.	May 10 2001	Jun 1 2001	156.00
2	10001	NuNuCa Nuß-Nougat-Creme	Mère Paillarde	May 13 2001	Jun 9 2001	420.00
3	10001	Boston Crab Meat	Mère Paillarde	May 13 2001	Jun 9 2001	736.00
4	10001	Raclette Courdavault	Mère Paillarde	May 13 2001	Jun 9 2001	440.00

	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
1	10000	Alice Mutton	Franchi S.p.A.	May 10 2001	Jun 1 2001	156.00
2	10001	NuNuCa Nuß-Nougat-Creme	Mère Paillarde	May 13 2001	Jun 9 2001	420.00
3	10001	Boston Crab Meat	Mère Paillarde	May 13 2001	Jun 9 2001	736.00
4	10001	Raclette Courdavault	Mère Paillarde	May 13 2001	Jun 9 2001	440.00

(383 row(s) affected)

Solution:

```sql
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 + 17, 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
```

Results:

```
/*
    Question 3
    3.  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 17 days t
    the product name from the product table, the customer name from the customer table, and
    Format the date order date and the shipped date as MON DD YYYY.  Use the formula (quant
    The query should produce the result set listed below.
*/

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 + 17, 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
```

100 %

Results | Messages

	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
1	10000	Alice Mutton	Franchi S.p.A.	May 10 2001	Jun 1 2001	156.00
2	10001	NuNuCa Nuß-Nougat-Creme	Mère Paillarde	May 13 2001	Jun 9 2001	420.00
3	10001	Boston Crab Meat	Mère Paillarde	May 13 2001	Jun 9 2001	736.00
4	10001	Raclette Courdavault	Mère Paillarde	May 13 2001	Jun 9 2001	440.00
5	10001	Wimmers gute Semmelknödel	Mère Paillarde	May 13 2001	Jun 9 2001	498.75
6	10002	Gorgonzola Telino	Folk och fä HB	May 14 2001	Jun 3 2001	437.50
7	10002	Chartreuse verte	Folk och fä HB	May 14 2001	Jun 3 2001	324.00
8	10002	Flotemysost	Folk och fä HB	May 14 2001	Jun 3 2001	322.50
9	10003	Camarvon Tigers	Simons bistro	May 15 2001	Jun 10 2001	750.00
10	10004	Thüringer Rostbratwurst	Vaffeljernet	May 16 2001	Jun 6 2001	4332.65
11	10004	Vegie-spread	Vaffeljernet	May 16 2001	Jun 6 2001	263.40
12	10005	Tarte au sucre	Wartian Herkku	May 20 2001	Jun 10 2001	295.80
13	10006	Konbu	Franchi S.p.A.	May 21 2001	Jun 10 2001	60.00
14	10006	Valkoinen suklaa	Franchi S.p.A.	May 21 2001	Jun 10 2001	65.00
15	10007	Queso Manchego La Pastora	Morgenstern Gesundkost	May 22 2001	Jun 28 2001	152.00
16	10007	Perth Pasties	Morgenstern Gesundkost	May 22 2001	Jun 28 2001	984.00
17	10007	Vegie-spread	Morgenstern Gesundkost	May 22 2001	Jun 28 2001	878.00
18	10008	Tofu	Furia Bacalhau e Frutos do Mar	May 23 2001	Jun 15 2001	465.00
19	10008	Sir Rodney's Scones	Furia Bacalhau e Frutos do Mar	May 23 2001	Jun 15 2001	150.00
20	10008	Manjimup Dried Apples	Furia Bacalhau e Frutos do Mar	May 23 2001	Jun 15 2001	1060.00
21	10009	Tunnbröd	Seven Seas Imports	May 24 2001	Jun 17 2001	630.00
22	10009	Manjimup Dried Apples	Seven Seas Imports	May 24 2001	Jun 17 2001	1590.00
23	10010	Ipoh Coffee	Simons bistro	May 28 2001	Jun 16 2001	460.00
24	10010	Flotemysost	Simons bistro	May 28 2001	Jun 16 2001	215.00
25	10011	Camarvon Tigers	Wellington Importadora	May 29 2001	Jun 20 2001	750.00

Query executed... | DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 383 rows

Step 4

Question:

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.

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

	customer_id	name	phone	order_id	order_date
1	BLAUS	Blauer See Delikatessen	0621-08460	11058	2004-03-23 00:00:00.000
2	BONAP	Bon app'	91.24.45.40	11076	2004-03-30 00:00:00.000
3	BOTTM	Bottom-Dollar Markets	(604) 555-4729	11045	2004-03-17 00:00:00.000
4	CACTU	Cactus Comidas para llevar	(1) 135-5555	11054	2004-03-22 00:00:00.000
5	ERNSH	Ernst Handel	7675-3425	11008	2004-03-02 00:00:00.000

(21 row(s) affected)

Solution:

```
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
```

Results:

```
/*
    Question 4
    4.  List all the orders that have not been shipped.
    Display the customer id, name and phone number from the customers table,
    Order the result set by the customer name.  The query should produce the
*/

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
```

100 %

Results | Messages

	customer_id	name	phone	order_id	order_date
1	BLAUS	Blauer See Delikatessen	0621-08460	11058	2004-03-23 00:00:00.000
2	BONAP	Bon app'	91.24.45.40	11076	2004-03-30 00:00:00.000
3	BOTTM	Bottom-Dollar Markets	(604) 555-4729	11045	2004-03-17 00:00:00.000
4	CACTU	Cactus Comidas para llevar	(1) 135-5555	11054	2004-03-22 00:00:00.000
5	ERNSH	Ernst Handel	7675-3425	11008	2004-03-02 00:00:00.000
6	ERNSH	Ernst Handel	7675-3425	11072	2004-03-29 00:00:00.000
7	GREAL	Great Lakes Food Market	(503) 555-7555	11061	2004-03-24 00:00:00.000
8	GREAL	Great Lakes Food Market	(503) 555-7555	11040	2004-03-16 00:00:00.000
9	LAMAI	La maison d'Asie	61.77.61.10	11051	2004-03-21 00:00:00.000
10	LEHMS	Lehmanns Marktstand	069-0245984	11070	2004-03-29 00:00:00.000
11	LILAS	LILA-Supermercado	(9) 331-6954	11071	2004-03-29 00:00:00.000
12	LILAS	LILA-Supermercado	(9) 331-6954	11065	2004-03-25 00:00:00.000
13	LINOD	LINO-Delicateses	(8) 34-56-12	11039	2004-03-15 00:00:00.000
14	PERIC	Pericles Comidas clásicas	(5) 552-3745	11073	2004-03-29 00:00:00.000
15	QUEEN	Queen Cozinha	(11) 555-1189	11068	2004-03-28 00:00:00.000
16	RANCH	Rancho grande	(1) 123-5555	11019	2004-03-07 00:00:00.000
17	RATTC	Rattlesnake Canyon Grocery	(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

DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 21 rows

Step 5

Question:

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)

Solution:

```
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
```

Results:

```
/*
    Question 5
    List all the customers where the region is NULL.
    Display the customer id, name, and city from the customers table, and the title d
    The query should produce the result set listed below.
*/

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
```

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
7	BLONP	Blondel père et fils	Strasbourg	Marketing Manager
8	BOLID	Bólido Comidas preparadas	Madrid	Owner
9	BONAP	Bon app'	Marseille	Owner
10	BSBEV	B's Beverages	London	Sales Representative
11	CACTU	Cactus Comidas para llevar	Buenos Aires	Sales Agent
12	CENTC	Centro comercial Moctezuma	México D.F.	Marketing Manager
13	CHOPS	Chop-suey Chinese	Bern	Owner
14	CONSH	Consolidated Holdings	London	Sales Representative
15	DRACD	Drachenblut Delikatessen	Aachen	Order Administrator
16	DUMON	Du monde entier	Nantes	Owner
17	EASTC	Eastern Connection	London	Sales Agent
18	ERNSH	Ernst Handel	Graz	Sales Manager
19	FISSA	FISSA Fabrica Inter. Salchichas S.A.	Madrid	Accounting Manager
20	FOLIG	Folies gourmandes	Lille	Assistant Sales Agent
21	FOLKO	Folk och fä HB	Bräcke	Owner
22	FRANK	Frankenversand	München	Marketing Manager
23	FRANR	France restauration	Nantes	Marketing Manager
24	FRANS	Franchi S.p.A.	Torino	Sales Representative
25	FURIB	Furia Bacalhau e Frutos do Mar	Lisboa	Sales Manager

Query exe... | DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 60 rows

Step 6

Question:

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.

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)

Solution:

```
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 supplier_name
GO
```

Results:

```
/*
    Question 6
    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, an
    Order the result set by the supplier name.
*/

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 supplier_name
GO
```

100 %

Results | Messages

	supplier_name	products_name	reorder_level	quantity_in_stock
1	Armstrong Company	Queso Cabrales	30	22
2	Cadbury Products Ltd.	Ipoh Coffee	25	17
3	Cadbury Products Ltd.	Røgede sild	15	5
4	Campbell Company	Gnocchi di nonna Alice	30	21
5	Dare Manufacturer Ltd.	Scottish Longbreads	15	6
6	Dare Manufacturer Ltd.	Sir Rodney's Scones	5	3
7	Edward's Products Ltd.	Chang	25	17
8	Edward's Products Ltd.	Aniseed Syrup	25	13
9	Kaplan Ltd.	Nord-Ost Matjeshering	15	10
10	New Orlean's Spices Ltd.	Louisiana Hot Spiced Okra	20	4
11	Ovellette Manufacturer Company	Chocolade	25	15
12	South Harbour Products Ltd.	Maxilaku	15	10
13	South Harbour Products Ltd.	Wimmers gute Semmelknödel	30	22
14	St. Jean's Company	Gorgonzola Telino	20	0
15	St. Jean's Company	Mascarpone Fabioli	25	9
16	Steveston Export Company	Gravad lax	25	11
17	Steveston Export Company	Outback Lager	30	15
18	Yves Delorme Ltd.	Longlife Tofu	5	4

Query executed s... | DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 18 rows

Step 7

Question:

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)

Solution:

```
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
```

Results:

```
/*
    Question 7
    Calculate the length in years from January 1, 2008 and when an order was shipped where
    Display the order id, and the shipped date from the orders table, the customer name, an
    Display the shipped date in the format MMM DD YYYY.
    Order the result set by order id and the calculated years.
*/

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
```

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
7	10006	Franchi S.p.A.	Paolo Accorti	May 24 2001	7
8	10007	Morgenstern Gesundkost	Alexander Feuer	Jun 11 2001	7
9	10008	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	May 29 2001	7
10	10009	Seven Seas Imports	Hari Kumar	May 31 2001	7
11	10010	Simons bistro	Jytte Petersen	May 30 2001	7
12	10011	Wellington Importadora	Paula Parente	Jun 3 2001	7
13	10012	LINO-Delicateses	Felipe Izquierdo	Jun 3 2001	7
14	10013	Richter Supermarkt	Michael Holz	Jun 7 2001	7
15	10014	GROSELLA-Restaurante	Manuel Pereira	Jun 12 2001	7
16	10015	Piccolo und mehr	Georg Pipps	Jun 20 2001	7
17	10016	Folies gourmandes	Martine Rancé	Jul 11 2001	7
18	10017	Blondel père et fils	Frédérique Citeaux	Jun 10 2001	7
19	10018	Rattlesnake Canyon Grocery	Paula Wilson	Jul 5 2001	7
20	10019	Magazzini Alimentari Riuniti	Giovanni Rovelli	Jun 20 2001	7
21	10020	Vins et alcools Chevalier	Paul Henriot	Jun 26 2001	7
22	10021	Ernst Handel	Roland Mendel	Jul 2 2001	7
23	10022	La maison d'Asie	Annette Roulet	Jul 22 2001	7
24	10023	Toms Spezialitäten	Karin Josephs	Jun 28 2001	7
25	10024	Rattlesnake Canyon Grocery	Paula Wilson	Jun 21 2001	7

Query executed... | DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 1057 rows

Step 8

Question:

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)
```

Solution:

```
SELECT
       'name' = LEFT(name, 1),
       'total' = COUNT(name)
FROM customers
GROUP BY LEFT(name, 1)
HAVING COUNT(name) >= 2 AND LEFT(name, 1) != 'S'
GO
```

Results:

```
/*
    Question 8
    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
*/

SELECT
'name' = LEFT(name, 1),
'total' = COUNT(name)
FROM customers
GROUP BY LEFT(name, 1)
HAVING COUNT(name) >= 2 AND LEFT(name, 1) != 'S'
GO
```

100 %

Results | Messages

	name	total
1	A	4
2	B	7
3	C	5
4	D	3
5	E	2
6	F	8
7	G	5
8	H	4
9	L	9
10	M	4
11	O	3
12	P	4
13	Q	3
14	R	6
15	T	6
16	V	3
17	W	5

Query executed s... | DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 17 rows

Step 9

Question:

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, the supplier_id and reorder level from the products 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)

Solution:

```sql
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
```

Results:

```sql
/*
    Question 9
    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, the sup
    Order the result set by the order id.
*/

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
```

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 s... | DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 15 rows

Step 10

Question:

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.

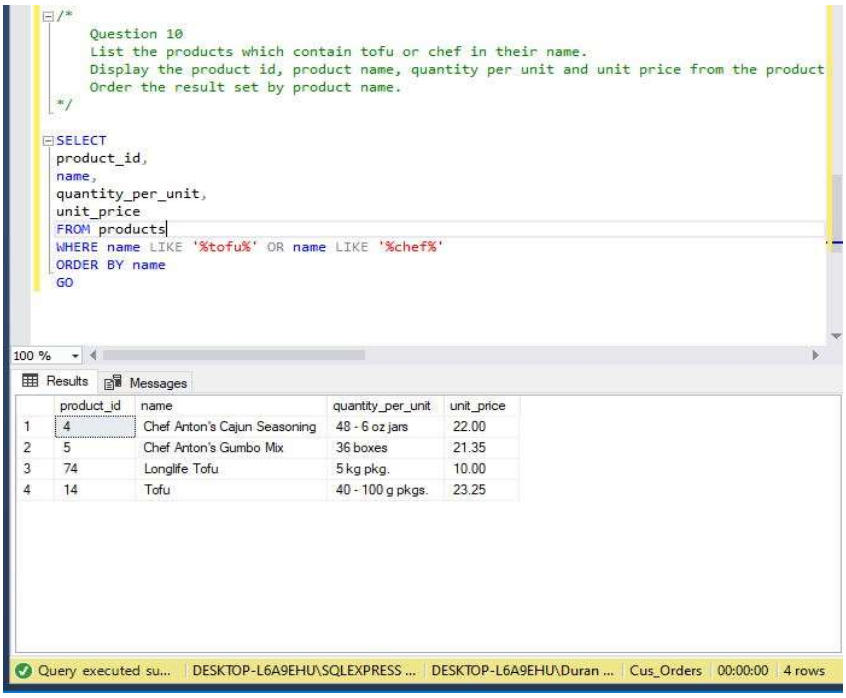
product_id	name	quantity_per_unit	unit_price
4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	22.0000
5	Chef Anton's Gumbo Mix	36 boxes	21.3500
74	Longlife Tofu	5 kg pkg.	10.0000
14	Tofu	40 - 100 g pkgs.	23.2500

(4 row(s) affected)

Solution:

```
SELECT
product_id,
name,
quantity_per_unit,
unit_price
FROM products
WHERE name LIKE '%tofu%' OR name LIKE '%chef%'
ORDER BY name
GO
```

Results:



Part C – INSERT, UPDATE, DELETE and VIEWS Statements

Step 1

Question:

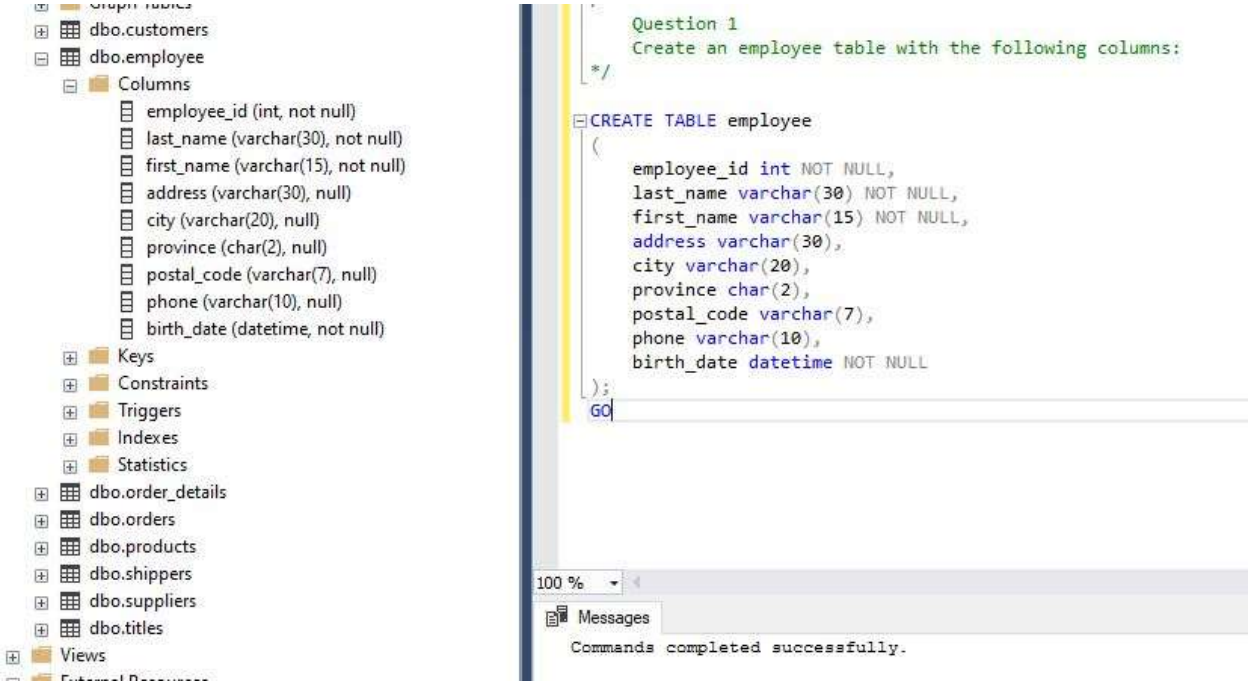
Create an **employee** table with the following columns:

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

Solution:

```
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:



Step 2

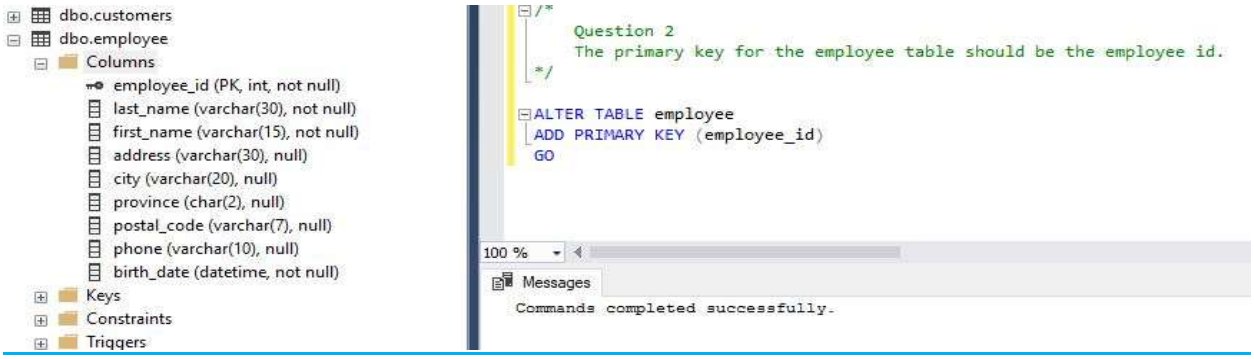
Question:

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

Solution:

```
ALTER TABLE employee
ADD PRIMARY KEY (employee_id)
GO
```

Results:



Step 3

Question:

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.

Solution:

```
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);
GO
```

Results:

```
/*
    Question 3
    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 e
*/

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);
GO
```

00 %

Messages

```
(9 rows affected)
```

Step 4

Question:

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

Solution:

```
INSERT INTO shippers(name)
VALUES('Quick Express')
GO

SELECT *
FROM shippers
GO
```

Results:

```
/*  Question 4
    Using the INSERT statement, add the shipper Quick Express to the shippers table.
*/

INSERT INTO shippers(name)
VALUES('Quick Express')
GO

SELECT *
FROM shippers
GO
```

100 %

Results | Messages

	shipper_id	name
1	1	Speedy Express
2	2	United Package
3	3	Federal Shipping
4	4	Quick Express

Step 5

Question:

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.

Solution:

```
UPDATE products
SET unit_price = unit_price * 1.05
WHERE unit_price >= 5 AND unit_price <= 10
GO
```

Results:

```
/*
    Question 5
    Using the UPDATE statement, increate the unit price in
*/

UPDATE products
SET unit_price = unit_price * 1.05
WHERE unit_price >= 5 AND unit_price <= 10
GO
```

100 %

Messages

```
(12 rows affected)
```

Step 6

Question:

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.

Solution:

```
UPDATE customers
SET fax = 'Unknown'
WHERE fax IS NULL
GO
```

Results:

```
/*
    Question 6
    Using the UPDATE statement, change the fax value to Unknown for all rows in th
*/

UPDATE customers
SET fax = 'Unknown'
WHERE fax IS NULL
GO
```

100 %

Messages

```
(22 rows affected)
```

Step 7

Question:

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)

Solution:

```
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
```

Results:

```
/*
    Question 7
    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 produc
    the customer name from the customers table, and the order cost.
    To calculate the cost of the orders, use the formula (order_details.c
    Run the view for the order ids between 10000 and 10200.
*/

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
8	10002	2001-05-14 00:00:00.000	71	Folk och fä HB	322.50
9	10003	2001-05-15 00:00:00.000	18	Simons bistro	750.00
10	10004	2001-05-16 00:00:00.000	29	Vaffeljemet	4332.65
11	10004	2001-05-16 00:00:00.000	63	Vaffeljemet	263.40
12	10005	2001-05-20 00:00:00.000	62	Wartian Herkku	295.80
13	10006	2001-05-21 00:00:00.000	13	Franchi S.p.A.	63.00
14	10006	2001-05-21 00:00:00.000	50	Franchi S.p.A.	65.00
15	10007	2001-05-22 00:00:00.000	12	Morgenstern Gesundkost	152.00
16	10007	2001-05-22 00:00:00.000	53	Morgenstern Gesundkost	984.00
17	10007	2001-05-22 00:00:00.000	63	Morgenstern Gesundkost	878.00
18	10008	2001-05-23 00:00:00.000	14	Furia Bacalhau e Frutos do Mar	465.00
19	10008	2001-05-23 00:00:00.000	21	Furia Bacalhau e Frutos do Mar	157.50
20	10008	2001-05-23 00:00:00.000	51	Furia Bacalhau e Frutos do Mar	1060.00
21	10009	2001-05-24 00:00:00.000	23	Seven Seas Imports	661.50
22	10009	2001-05-24 00:00:00.000	51	Seven Seas Imports	1590.00
23	10010	2001-05-28 00:00:00.000	43	Simons bistro	460.00

DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 540 rows

Step 8

Question:

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	1967.03.04
7	King, Robert	1972.05.29
9	Dodsworth, Anne	1978.01.27

(3 row(s) affected)

Solution:

```
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
```

Results:

```
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
```

100 %

Results | Messages

	employee_id	name	birth_date
1	5	Buchanan, Steven	1967.03.04
2	7	King, Robert	1972.05.29
3	9	Dodsworth, Anne	1978.01.27

Step 9

Question:

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
10256	WELLI	Wellington Importadora	Resende	Brazil	Jun 10 2002
10269	WHITC	White Clover Markets	Seattle	United States	Jul 3 2002
10344	WHITC	White Clover Markets	Seattle	United States	Sep 29 2002
10374	WOLZA	Wolski Zajazd	Warszawa	Poland	Nov 2 2002

(293 row(s) affected)

Solution:

```sql
CREATE VIEW vw_all_orders
AS
SELECT
       orders.order_id,
       orders.shipped_date,
       customers.customer_id,
       'customer_name' = customers.name,
       customers.city,
       customers.country
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
```

Results:

```
/*
    Question 9
    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 fro
    Run the view for orders shipped from January 1, 2002 and December 31, 2002, formatting the shipped date as MO
    Order the result set by customer name and country.
*/

CREATE VIEW vw_all_orders
AS
SELECT
    orders.order_id,
    orders.shipped_date,
    customers.customer_id,
    'customer_name' = customers.name,
    customers.city,
    customers.country
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
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
8	10383	AROUT	Around the Horn	London	United Kingdom	Nov 11 2002
9	10238	BSBEV	B's Beverages	London	United Kingdom	May 20 2002
10	10289	BSBEV	B's Beverages	London	United Kingdom	Jul 22 2002
11	10384	BERGS	Berglunds snabbköp	Luleå	Sweden	Nov 13 2002
12	10278	BERGS	Berglunds snabbköp	Luleå	Sweden	Jul 10 2002
13	10280	BERGS	Berglunds snabbköp	Luleå	Sweden	Aug 6 2002
14	10158	BERGS	Berglunds snabbköp	Luleå	Sweden	Feb 4 2002
15	10171	BERGS	Berglunds snabbköp	Luleå	Sweden	Feb 28 2002
16	10213	BERGS	Berglunds snabbköp	Luleå	Sweden	Apr 22 2002
17	10233	BERGS	Berglunds snabbköp	Luleå	Sweden	May 15 2002
18	10265	BLONP	Blondel père et fils	Strasbourg	France	Jul 6 2002
19	10297	BLONP	Blondel père et fils	Strasbourg	France	Aug 4 2002
20	10360	BLONP	Blondel père et fils	Strasbourg	France	Oct 26 2002
21	10326	BOLID	Bólido Comidas preparadas	Madrid	Spain	Sep 7 2002
22	10241	BOLID	Bólido Comidas preparadas	Madrid	Spain	May 27 2002
23	10200	BOLID	Bólido Comidas preparadas	Madrid	Spain	Apr 29 2002

Query executed successfully. | DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 293 rows

Step 10

Question:

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)

Solution:

```
CREATE VIEW vw_supplier_products_shipped
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_shipped
GO
```

Results:

```sql
/*
    Question 10
    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 tabl
    Run the view.
*/

CREATE VIEW vw_supplier_products_shipped
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_shipped
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
8	3	Macaulay Products Company	8	Northwoods Cranberry Sauce
9	4	Yves Delorme Ltd.	9	Mishi Kobe Niku
10	4	Yves Delorme Ltd.	10	Ikura
11	5	Amstrong Company	11	Queso Cabrales
12	5	Amstrong Company	12	Queso Manchego La Pastora
13	6	Catelli Products Ltd.	13	Konbu
14	6	Catelli Products Ltd.	14	Tofu
15	6	Catelli Products Ltd.	15	Genen Shouyu
16	7	Steveston Export Company	16	Pavlova
17	7	Steveston Export Company	17	Alice Mutton
18	7	Steveston Export Company	18	Camarvon Tigers
19	8	Dare Manufacturer Ltd.	19	Teatime Chocolate Biscuits
20	8	Dare Manufacturer Ltd.	20	Sir Rodney's Marmalade
21	8	Dare Manufacturer Ltd.	21	Sir Rodney's Scones
22	9	Silver Spring Wholesale Market	22	Gustaf's Knäckebröd
23	9	Silver Spring Wholesale Market	23	Tunnbröd

Query executed successfully. | DESKTOP-L6A9EHU\SQLEXPRESS ... | DESKTOP-L6A9EHU\Duran ... | Cus_Orders | 00:00:00 | 77 rows

Part D – Stored Procedures and Triggers

Step 1

Question:

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 Warhurst Rd.	London	(71) 555-1717

(6 row(s) affected)

Solution:

```
CREATE PROCEDURE sp_customer_city
(@city varchar(30))
AS
SELECT
       customer_id,
       name,
       address,
       city,
       phone
FROM customers
WHERE city = @city
GO

EXECUTE sp_customer_city 'London'
GO
```

Results:

```
/*
    Question 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.
*/

CREATE PROCEDURE sp_customer_city
(@city varchar(30))
AS
SELECT
customer_id,
name,
address,
city,
phone
FROM customers
WHERE city = @city
GO
EXECUTE sp_customer_city 'London'
GO
```

100 %

Results | Messages

	customer_id	name	address	city	phone
1	AROUT	Around the Hom	120 Hanover Sq.	London	(71) 555-7788
2	BSBEV	B's Beverages	Fauntleroy Circus	London	(71) 555-1212
3	CONSH	Consolidated Holdings	Berkeley Gardens 12 Brewery	London	(71) 555-2282
4	EASTC	Eastern Connection	35 King George	London	(71) 555-0297
5	NORTS	North/South	South House 300 Queensbridge	London	(71) 555-7733
6	SEVES	Seven Seas Imports	90 Wadhurst Rd.	London	(71) 555-1717

Step 2

Question:

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.

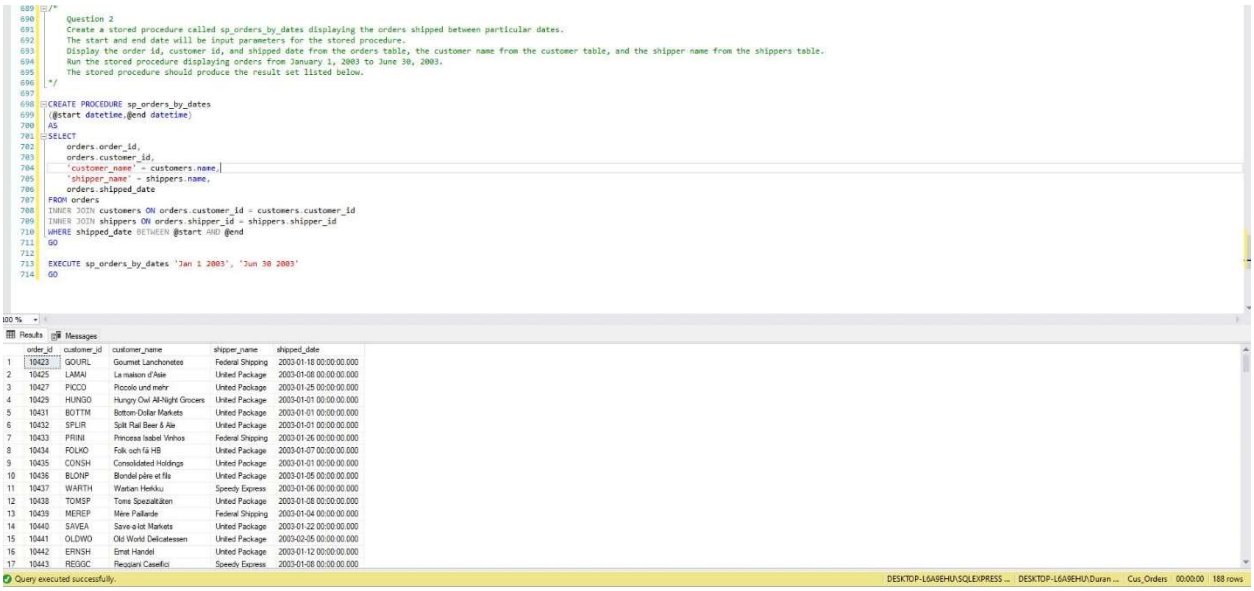
order_id	customer_id	customer_name	shipper_name	shipped_date
10423	GOURL	Gourmet Lanchonetes	Federal Shipping	2003-01-18 00:00:00.000
10425	LAMAI	La maison d'Asie	United Package	2003-01-08 00:00:00.000
10427	PICCO	Piccolo und mehr	United Package	2003-01-25 00:00:00.000
10429	HUNGO	Hungry Owl All-Night 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)

Solution:

```
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
```

Results:



	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
8	10434	FOLKO	Folk och fä HB	United Package	2003-01-07 00:00:00.000
9	10435	CONSH	Consolidated Holdings	United Package	2003-01-01 00:00:00.000
10	10436	BLONP	Blondel père et fils	United Package	2003-01-05 00:00:00.000
11	10437	WARTH	Wartian Herkku	Speedy Express	2003-01-06 00:00:00.000
12	10438	TOMSP	Toms Spezialitäten	United Package	2003-01-08 00:00:00.000
13	10439	MEREP	Mère Paillarde	Federal Shipping	2003-01-04 00:00:00.000
14	10440	SAVEA	Save-a-lot Markets	United Package	2003-01-22 00:00:00.000
15	10441	OLDWO	Old World Delicatessen	United Package	2003-02-05 00:00:00.000
16	10442	ERNSH	Ernst Handel	United Package	2003-01-12 00:00:00.000
17	10443	REGGC	Reggiani Caseifici	Speedy Express	2003-01-08 00:00:00.000

Step 3

Question:

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)

Solution:

```
CREATE PROCEDURE sp_product_listing
(@product varchar(50),@month varchar(8),@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
```

Results:

```
/*
3.  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.
*/
CREATE PROCEDURE sp_product_listing
(@product varchar(50),@month varchar(8),@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

Step 4

Question:

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:

Results | Messages

	Product_ID	Product Name	Quantity being deleted from Order	In stock Quantity after Deletion
1	25	NuNuCa Nuß-Nougat-Creme	30	106

Solution:

```
CREATE TRIGGER tr_order_details
ON order_details
AFTER DELETE
AS
DECLARE @prod_id intid, @qty_del int
SELECT @prod_id = product_id, @qty_del = quantity
FROM deleted
UPDATE products
SET quantity_in_stock = quantity_in_stock + @qty_del
WHERE product_id = @prod_id
BEGIN
SELECT
       'Product_ID' = deleted.product_id,
       'Product Name' = products.name,
       'Quantity being deleted from Order' = @qty_del,
       'In stock Quantity after Deletion' = products.quantity_in_stock
FROM deleted
INNER JOIN products ON deleted.product_id = products.product_id
END
GO

DELETE order_details
WHERE order_id = 10001 AND product_id = 25
       GO
```

Results:

```
/*
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
*/

CREATE TRIGGER tr_order_details
ON order_details
AFTER DELETE
AS
DECLARE @prod_id intid, @qty_del int
SELECT @prod_id = product_id, @qty_del = quantity
FROM deleted
UPDATE products
SET quantity_in_stock = quantity_in_stock + @qty_del
WHERE product_id = @prod_id
BEGIN
SELECT
     'Product_ID' = deleted.product_id,
     'Product Name' = products.name,
     'Quantity being deleted from Order' = @qty_del,
     'In stock Quantity after Deletion' = products.quantity_in_stock
FROM deleted
INNER JOIN products ON deleted.product_id = products.product_id
END
GO

DELETE order_details
WHERE order_id = 10001 AND product_id = 25
GO
```

100 %

Results | Messages

	Product_ID	Product Name	Quantity being deleted from Order	In stock Quantity after Deletion
1	25	NuNuCa Nuß-Nougat-Creme	30	106

Query executed successfully.

Step 5

Question:

Create an **UPDATE** trigger called **tr_qty_check** on the order_details table which will reject any update to the quantity column if an addition to the original quantity cannot be supplied from the existing quantity in stock. The trigger should also report on the additional quantity needed and the quantity available. If there is enough stock, the trigger should update the stock value in the products table by subtracting the additional quantity from the original stock value and display the updated stock value.

Solution:

```
CREATE TRIGGER tr_qty_check
ON order_details
FOR UPDATE
AS
DECLARE @prod_id intid, @gty int, @quantity_INSTOCK int
SELECT  @prod_id = products.product_id, @gty = inserted.quantity-deleted.quantity,
@quantity_INSTOCK = products.quantity_in_stock
FROM inserted
INNER JOIN deleted ON inserted.product_id = deleted.product_id
INNER JOIN products ON inserted.product_id = products.product_id
IF(@gty > @quantity_INSTOCK)
BEGIN
     PRINT 'additional quantity needed'
        ROLLBACK TRANSACTION
END
ELSE
BEGIN
   UPDATE products
   SET quantity_in_stock = quantity_in_stock - @gty
   WHERE product_id = @prod_id
END
GO
```

Results:

```
/*
    Question 5
    Create an UPDATE trigger called tr_qty_check on the order_details table which will reject any update to the quantity column if
    The trigger should also report on the additional quantity needed and the quantity available.
    If there is enough stock, the trigger should update the stock value in the products table by subtracting the additional quantit
*/

CREATE TRIGGER tr_qty_check
ON order_details
FOR UPDATE
AS
DECLARE @prod_id intid, @gty int, @quantity_INSTOCK int
SELECT  @prod_id = products.product_id, @gty = inserted.quantity-deleted.quantity, @quantity_INSTOCK = products.quantity_in_stock
FROM inserted
INNER JOIN deleted ON inserted.product_id = deleted.product_id
INNER JOIN products ON inserted.product_id = products.product_id
IF(@gty > @quantity_INSTOCK)
BEGIN
     PRINT 'additional quantity needed'
     ROLLBACK TRANSACTION
END
ELSE
BEGIN
   UPDATE products
   SET quantity_in_stock = quantity_in_stock - @gty
   WHERE product_id = @prod_id
END
GO
```

100 %

Messages

```
Commands completed successfully.
```

Step 6

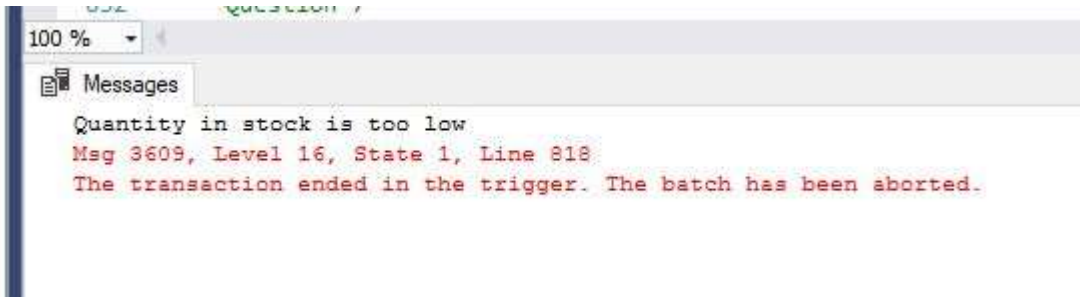
Question:

Run the following 2 queries separately to verify your trigger:

Solution:

```
UPDATE order_details
SET quantity =50
WHERE order_id = '10044'
   AND product_id = 7;

UPDATE order_details
SET quantity =40
WHERE order_id = '10044'
   AND product_id = 7;
```

Results:



Step 7

Question:

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

Solution:

```
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:

```
/*
    Question 7
    Create a stored procedure called sp_del_inactive_cust to delete customers that have no orders.
    The stored procedure should delete 1 row.
*/

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
```

100 %

Messages

```
(1 row affected)
```

Step 8

Question:

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.

Solution:

```
CREATE PROCEDURE sp_employee_information
( @employ_id int )
AS
SELECT
        employee_id,
        last_name,
        first_name,
        address,
        city,
        province,
        postal_code,
        phone,
        birth_date
FROM employee
WHERE employee_id = @employ_id
GO

EXECUTE sp_employee_information 5
GO
```

Results:

```
/*
    Question 8
    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.
*/

CREATE PROCEDURE sp_employee_information
( @employ_id int )
AS
SELECT
    employee_id,
    last_name,
    first_name,
    address,
    city,
    province,
    postal_code,
    phone,
    birth_date
FROM employee
WHERE employee_id = @employ_id

GO
EXECUTE sp_employee_information 5
GO
```

100 %

Results | Messages

	employee_id	last_name	first_name	address	city	province	postal_code	phone	birth_date
1	5	Buchanan	Steven	14 Garrett Hill	New Westminster	BC	V1G 8J7	6045554848	1967-03-04 00:00:00.000

Step 9

Question:

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.

Solution:

```
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
```

Results:

```
/*
    Question 9
    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
*/

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
```

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	7	Macaulay Products Company	4800 Kingsway	Burnaby	BC	3	10
5	11	Armstrong Company	1638 Derwent Way	Richmond	BC	22	30
6	17	Steveston Export Company	2951 Moncton Street	Richmond	BC	0	0
7	21	Dare Manufacturer Ltd.	1603 3rd Avenue West	Burnaby	BC	3	5
8	29	South Harbour Products Ltd.	35 Yale Crescent	Surrey	BC	0	0
9	30	Kaplan Ltd.	3016 19th Street South	Vancouver	BC	10	15
10	31	St. Jean's Company	119 Cowley Road	Burnaby	BC	0	20
11	32	St. Jean's Company	119 Cowley Road	Burnaby	BC	9	25
12	37	Steveston Export Company	2951 Moncton Street	Richmond	BC	11	25
13	38	Dare Manufacturer Ltd.	1603 3rd Avenue West	Burnaby	BC	17	15
14	43	Cadbury Products Ltd.	12840 Trites	Vancouver	BC	17	25
15	45	Cadbury Products Ltd.	12840 Trites	Vancouver	BC	5	15
16	48	Ovellette Manufacturer Company	272 Gladstone Avenue	Delta	BC	15	25
17	49	South Harbour Products Ltd.	35 Yale Crescent	Surrey	BC	10	15
18	53	St. Jean's Company	119 Cowley Road	Burnaby	BC	0	0
19	56	Campbell Company	892 Farrell Avenue	New Westminster	BC	21	30
20	64	South Harbour Products Ltd.	35 Yale Crescent	Surrey	BC	22	30
21	66	New Orlean's Spices Ltd.	1040 Georgia Street West	Vancouver	BC	4	20
22	68	Dare Manufacturer Ltd.	1603 3rd Avenue West	Burnaby	BC	6	15
23	70	Steveston Export Company	2951 Moncton Street	Richmond	BC	15	30
24	74	Yves Delorme Ltd.	3050 Granville Street	New Westminster	BC	4	5

Step 10

Question:

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.

Solution:

```
CREATE PROCEDURE sp_unit_prices
(
@unit_1 money,
@unit_2 money
)
AS
SELECT
```

```
        product_id, name,
        alternate_name,
        unit_price
FROM products
WHERE unit_price BETWEEN @unit_1 AND @unit_2
GO

EXECUTE sp_unit_prices 5, 10
GO
```

Results:

```
/*
    Question 10
    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.
*/

CREATE PROCEDURE sp_unit_prices
(
@unit_1 money,
@unit_2 money
)
AS
SELECT
    product_id, name,
    alternate_name,
    unit_price
FROM products
WHERE unit_price BETWEEN @unit_1 AND @unit_2
GO

EXECUTE sp_unit_prices 5, 10
GO
```

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

3. CONCLUSION

The project has been great learning experience about everything I have learned in COMP 1630. The strength and flexibility of the SQL language was entertaining to work though and see positive results being achieved. All questions have been completed however last part was very challenging.

4. SQL SCRIPTS

```
-- Part A - Database and Tables

/*
   Question 1

   Create a database called Cus_Orders.

*/

USE master
GO

CREATE DATABASE  Cus_Orders
GO

USE Cus_Orders
GO

/*
       Question 2

       2.     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.
*/

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

/*
       Question 3

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

              customers
              orders
              order_details
              products
              shippers
              suppliers
              titles

*/

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)
);

CREATE TABLE orders
(
        order_id intid,
        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
);

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) NOT NULL
);

CREATE TABLE suppliers
(
```

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

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

/*
	Question 4
	Set the primary keys and foreign keys for the tables.
*/

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_cust_titles FOREIGN KEY (title_id)
REFERENCES titles(title_id);

ALTER TABLE orders
ADD CONSTRAINT fk_orders_cust 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_order_details_orders FOREIGN KEY (order_id)
REFERENCES orders(order_id);
```

```sql
ALTER TABLE order_details
ADD CONSTRAINT fk_order_details_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

/*
       Question 5

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
*/

ALTER TABLE customers
ADD CONSTRAINT default_country DEFAULT('Canada') FOR country;

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

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

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

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

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

GO


/*
       Question 6
       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)

*/

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'
         )
```

```sql
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'
         )



--             Part B - SQL Statements

/*
       Question 1
List the customer id, name, city, and country from the customer table.
Order the result set by the customer id.
*/

SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
GO

/*
       Question 2
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.
*/

ALTER TABLE customers
ADD active BIT NOT NULL
CONSTRAINT default_active DEFAULT(1);
GO

/*
       Question 3
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 17 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.
*/
```

```sql
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 + 17, 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

/*
	Question 4
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.
*/

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

/*
	Question 5
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.
*/

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

/*
	Question 6
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.
*/

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 supplier_name
GO

/*
       Question 7
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.
*/

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


/*
       Question 8
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.
*/

SELECT
'name' = LEFT(name, 1),
'total' = COUNT(name)
FROM customers
GROUP BY LEFT(name, 1)
HAVING COUNT(name) >= 2 AND LEFT(name, 1) != 'S'
GO

/*
       Question 9
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, the
supplier_id and reorder level from the products table.
```

```
Order the result set by the order id.
*/

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

/*
	Question 10
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.
*/

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

/*
	Question 1
Create an employee table with the following columns:
*/

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

/*
	Question 2
```

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

ALTER TABLE employee
ADD PRIMARY KEY (employee_id)
GO

/*
	Question 3
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.
*/

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);
GO

/*	Question 4
Using the INSERT statement, add the shipper Quick Express to the shippers table.
*/

INSERT INTO shippers(name)
VALUES('Quick Express')
GO

SELECT *
FROM shippers
GO

/*
	Question 5
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.
*/

UPDATE products
SET unit_price = unit_price * 1.05
WHERE unit_price >= 5 AND unit_price <= 10
GO

/*
	Question 6
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.
*/
```

```sql
UPDATE customers
SET fax = 'Unknown'
WHERE fax IS NULL
GO

/*
	Question 7
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 table, 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.
*/

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

/*
	Question 8
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.
*/

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
```

```sql
/*
	Question 9
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.
*/

CREATE VIEW vw_all_orders
AS
SELECT
	orders.order_id,
	orders.shipped_date,
	customers.customer_id,
	'customer_name' = customers.name,
	customers.city,
	customers.country
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

/*
	Question 10
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.
*/

CREATE VIEW vw_supplier_products_shipped
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_shipped
```

```sql
GO

-- PART D Stored Procedures and Triggers

/*
	Question 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.
*/

CREATE PROCEDURE sp_customer_city
(@city varchar(30))
AS
SELECT
customer_id,
name,
address,
city,
phone
FROM customers
WHERE city = @city
GO
EXECUTE sp_customer_city 'London'
GO

/*
	Question 2
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.
*/

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
```

```
/*
	Question 3
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.
*/
CREATE PROCEDURE sp_product_listing
(@product varchar(50),@month varchar(8),@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

/*
	Question 4
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
*/

CREATE TRIGGER tr_order_details
ON order_details
AFTER DELETE
AS
DECLARE @prod_id intid, @qty_del int
SELECT @prod_id = product_id, @qty_del = quantity
FROM deleted
UPDATE products
SET quantity_in_stock = quantity_in_stock + @qty_del
WHERE product_id = @prod_id
BEGIN
SELECT
	'Product_ID' = deleted.product_id,
	'Product Name' = products.name,
	'Quantity being deleted from Order' = @qty_del,
	'In stock Quantity after Deletion' = products.quantity_in_stock
```

```
FROM deleted
INNER JOIN products ON deleted.product_id = products.product_id
END
GO

DELETE order_details
WHERE order_id = 10001 AND product_id = 25
GO

/*
       Question 5
Create an UPDATE trigger called tr_qty_check on the order_details table which will reject
any update to the quantity column if an addition to the original quantity cannot be
supplied from the existing quantity in stock.
The trigger should also report on the additional quantity needed and the quantity
available.
If there is enough stock, the trigger should update the stock value in the products table
by subtracting the additional quantity from the original stock value and display the
updated stock value.
*/

CREATE TRIGGER tr_qty_check
ON order_details
FOR UPDATE
AS
DECLARE @prod_id intid, @gty int, @quantity_INSTOCK int
SELECT  @prod_id = products.product_id, @gty = inserted.quantity-deleted.quantity,
@quantity_INSTOCK = products.quantity_in_stock
FROM inserted
INNER JOIN deleted ON inserted.product_id = deleted.product_id
INNER JOIN products ON inserted.product_id = products.product_id
IF(@gty > @quantity_INSTOCK)
BEGIN
     PRINT 'additional quantity needed'
        ROLLBACK TRANSACTION
END
ELSE
BEGIN
   UPDATE products
   SET quantity_in_stock = quantity_in_stock - @gty
   WHERE product_id = @prod_id
END
GO


/*
       Question 6
Run the following 2 queries separately to verify your trigger:un the following 2 queries
separately to verify your trigger:
*/

UPDATE order_details
SET quantity =50
WHERE order_id = '10044'
     AND product_id = 7;

UPDATE order_details
```

```
SET quantity =40
WHERE order_id = '10044'
     AND product_id = 7;


/*
       Question 7
Create a stored procedure called sp_del_inactive_cust to delete customers that have no
orders.
The stored procedure should delete 1 row.
*/

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

/*
       Question 8
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.
*/

CREATE PROCEDURE sp_employee_information
( @employ_id int )
AS
SELECT
       employee_id,
       last_name,
       first_name,
       address,
       city,
       province,
       postal_code,
       phone,
       birth_date
FROM employee
WHERE employee_id = @employ_id
GO

EXECUTE sp_employee_information 5
GO
```

```
/*
	Question 9
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
*/

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
/*
	Question 10
	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.
*/
CREATE PROCEDURE sp_unit_prices
(
@unit_1 money,
@unit_2 money
)
AS
SELECT
	product_id, name,
	alternate_name,
	unit_price
FROM products
WHERE unit_price BETWEEN @unit_1 AND @unit_2
GO

EXECUTE sp_unit_prices 5, 10
GO
```