PROJECT-2

COMP-1630 Relational Database & SQL

By EMRE PACACI

Instructor: Mark Bacchus

Table of Contents

1. INTRODUCTION

In this project, **Cus_Orders** database has been created and manipulated using Microsoft SQL Server Management Studio. Each question is listed, and SQL scripts have been written to execute based on those questions. Also, taken a screenshot to indicate successful results.

2. SOLUTIONS

Part A - Database and Table

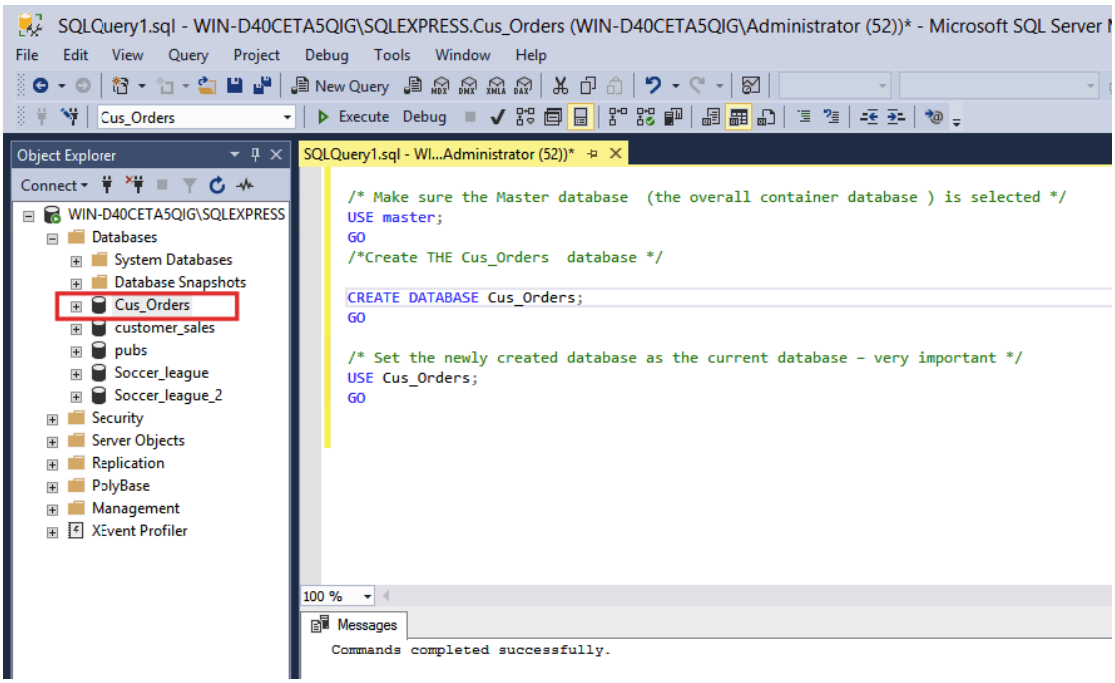
Question 1:

Create a database called **Cus_Orders.**

SQL statements used:

```
 /* Make sure the Master database (the overall container
database) is selected */
 USE master;
 GO
/*Create THE Cus_Orders  database */
 CREATE DATABASE Cus_Orders;
 GO
/* Set the newly created database as the current database –
very important */
 USE Cus_Orders;
 GO
```

Producing the following results:



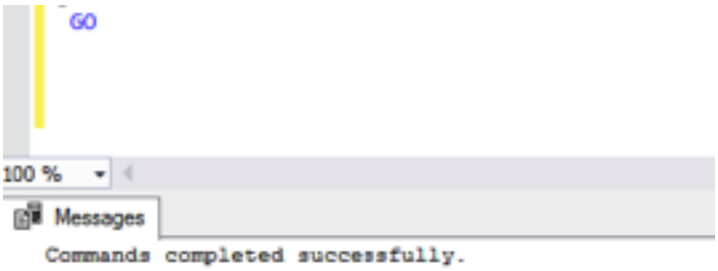
Question 2:

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.

SQL statements used:

```
/* create 2 user defined data types for customer_id,
        product_id, order_id, etc as int and char*/
CREATE TYPE custtid FROM char(5) NOT NULL;
CREATE TYPE inttid FROM int NOT NULL;
GO
```

Producing the following results:



Question 3:

Create the following tables; **customers**, **orders, order_details, products, shippers, supplier, titles**

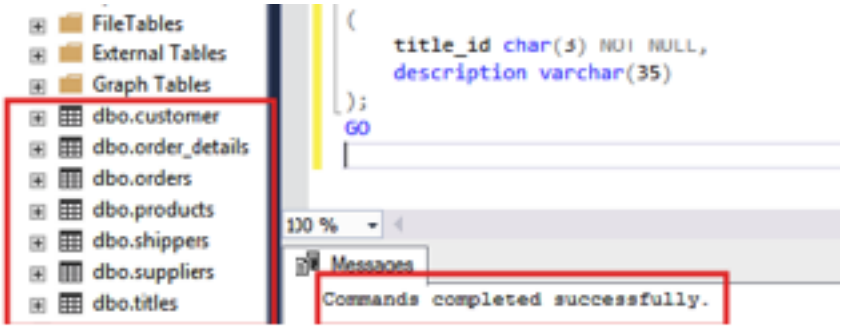
SQL statements used:

```
/* Create Customers table */
 CREATE TABLE customers
 (
    customer_id custtid,
    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 orders table*/
 CREATE TABLE orders
 (
    order_id inttid,
    customer_id custtid,
    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 order_details table*/
CREATE TABLE order_details
(
   order_id inttid,
   product_id inttid,
   quantity int NOT NULL,
   discount float NOT NULL
);
/* Create products table */
CREATE TABLE products
(
   product_id inttid,
   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 shippers table*/
CREATE TABLE shippers
```

```
 (
    shipper_id int IDENTITY NOT NULL,
    name varchar(20));
 /* Create suppliers table */
 CREATE TABLE suppliers
 (
    supplier_id int IDENTITY NOT NULL,
    name varchar(40) NOT NULL,
    address varchar(30),
    city varchar(20),
    province char(2)
 );
/* Create titles table */
CREATE TABLE titles
(
    title_id char(3) NOT NULL,
    description varchar(35)
);
GO
```

Producing the following results:



Question 4:

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

SQL statements used:

```sql
ALTER TABLE customers
ADD PRIMARY KEY (customer_id);

ALTER TABLE orders
ADD PRIMARY KEY (order_id);

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

ALTER TABLE products
ADD PRIMARY KEY (product_id);

ALTER TABLE shippers
ADD PRIMARY KEY (shipper_id);

ALTER TABLE suppliers
ADD PRIMARY KEY (supplier_id);

ALTER TABLE titles
ADD PRIMARY KEY (title_id);

GO

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

ALTER TABLE orders
ADD CONSTRAINT fk_order_cus FOREIGN KEY(customer_id)
REFERENCES customers(customer_id);
```

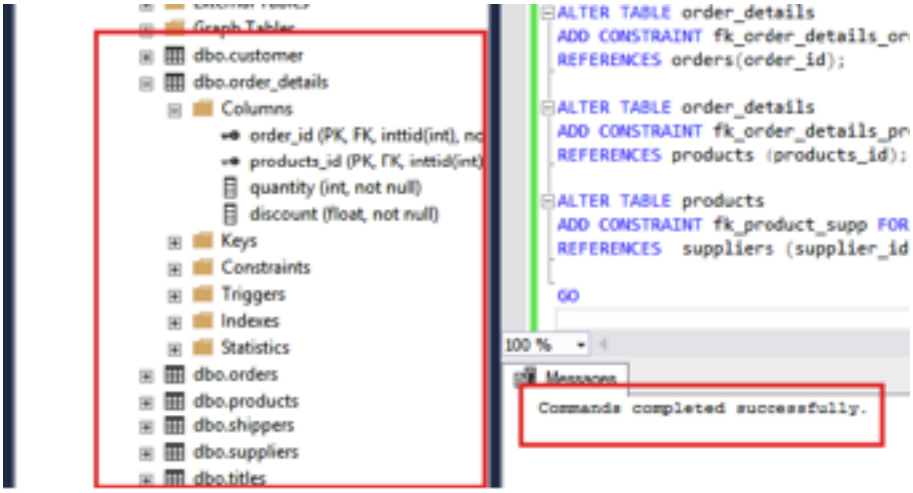
```
ALTER TABLE orders
ADD CONSTRAINT fk_order_shipper FOREIGN KEY (ship-
per_id)
REFERENCES shippers (shipper_id);

ALTER TABLE order_details
ADD CONSTRAINT fk_order_details_order FOREIGN KEY(or-
der_id)
REFERENCES orders(order_id);

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

ALTER TABLE products
ADD CONSTRAINT fk_product_supp FOREIGN KEY(sup-
plier_id)
REFERENCES  suppliers (supplier_id)
GO.
```

Producing the following results:



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

SQL statements used:

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

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

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

ALTER TABLE products
ADD CONSTRAINT check_reorder_level CHECK (reor-
der_level >= 1);

ALTER TABLE products
ADD CONSTRAINT check_quantity_in_stoct CHECK (quan-
tity_in_stock < 150);

ALTER TABLE suppliers \ADD CONSTRAINT default_province
DEFAULT ('BC') FOR provinc
```

Producing the following results:

```
ALTER TABLE products
ADD CONSTRAINT check_quantity_in_stoct CHECK (quantity_in_stoc

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

100 %

Messages

Commands completed successfully.

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)

Expected results:

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)

SQL statements used:

```sql
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'
       )
```

```sql
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
```

Producing the following results:

```
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

Question-1:

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

Expected results:

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)

SQL statements used:

```sql
SELECT customer_id,
       name,
       city,
       country
FROM customers
ORDER BY customer_id

GO
```

Producing the following results:

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

Qu | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 91 rows

100 %

Results | Messages

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

Qu | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 91 rows

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**.

SQL statement used:

```
--Question 2

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

Producing the following result:

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

100 %

Messages

Commands completed successfully.

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.

Expected results:

	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
1	10000	Alice Mutton	Franchi S.p.A.	May 10 2001	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
379	10137	Scottish Longbreads	Antonio More...	Dec 26 2001	Feb 8 2002	187.50
380	10137	Mozzarella di Giovanni	Antonio More...	Dec 26 2001	Feb 8 2002	870.00
381	10138	Inlagd Sill	Du monde ent...	Dec 27 2001	Jan 20 2002	228.00
382	10138	Louisiana Hot Spiced Okra	Du monde ent...	Dec 27 2001	Jan 20 2002	204.00
383	10139	Camembert Pierrot	Vaffeljernet	Dec 30 2001	Jan 26 2002	680.00

SQL statement used:

```sql
-- Question 3

SELECT  orders.order_id,
        'product_name' = products.name,
        'customer_name' = customers.name,
        'order_date'= CONVERT(char(11), orders.order_date, 100),
        'new_shipped_date' = CONVERT(char(11), DATEADD(day, 17,  orders.shipped_date), 100),
        'order_cost' = (order_details.quantity * products.unit_price)
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN order_details ON order_details.order_id = orders.order_id
INNER JOIN products ON products.product_id = order_details.product_id
WHERE order_date BETWEEN 'January 1, 2001' AND  'December 31, 2001'
```

Producing the following result:

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	Fløtemysost	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

Query exec... | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 383 rows

…

100 %

Results | Messages

	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
374	10133	Gnocchi di nonna Alice	Reggiani Cas...	Dec 19 2001	Jan 9 2002	456.00
375	10134	Filo Mix	Wartian Herkku	Dec 20 2001	Jan 13 2002	175.00
376	10135	Chartreuse verte	GROSELLA-...	Dec 24 2001	Jan 20 2002	90.00
377	10136	Camarvon Tigers	Hungry Owl Al...	Dec 25 2001	Jan 16 2002	1500.00
378	10137	Konbu	Antonio More...	Dec 26 2001	Feb 8 2002	120.00
379	10137	Scottish Longbreads	Antonio More...	Dec 26 2001	Feb 8 2002	187.50
380	10137	Mozzarella di Giovanni	Antonio More...	Dec 26 2001	Feb 8 2002	870.00
381	10138	Inlagd Sill	Du monde ent...	Dec 27 2001	Jan 20 2002	228.00
382	10138	Louisiana Hot Spiced Okra	Du monde ent...	Dec 27 2001	Jan 20 2002	204.00
383	10139	Camembert Pierrot	Vaffeljernet	Dec 30 2001	Jan 26 2002	680.00

Query exec... | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 383 rows

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.

Expected results:

	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

...

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

SQL statements used:

```
--Question 4

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

Producing the following result:

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 21 rows

..

100 %

Results | Messages

	customer_id	name	phone	order_id	order_date
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 Gro...	(505) 555-5939	11077	2004-03-30 00:00:00.000
18	REGGC	Reggiani Caseifici	0522-556721	11062	2004-03-24 00:00:00.000
19	RICAR	Ricardo Adocicados	(21) 555-3412	11059	2004-03-23 00:00:00.000
20	RICSU	Richter Supermarkt	0897-034214	11075	2004-03-30 00:00:00.000
21	SIMOB	Simons bistro	31 12 34 56	11074	2004-03-30 00:00:00.000

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 21 rows

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.

Expected results:

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)

SQL statements used:

```sql
--Question 5

SELECT customers.customer_id,
       customers.name,
       customers.city,
       titles.description
FROM customers
INNER JOIN titles ON customers.title_id = titles.title_id
WHERE customers.region is NULL
```

Producing the following results:

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 60 rows

..

100 %

Results | Messages

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 60 rows

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.

Expected results:

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

(18 row(s) affected)

SQL statements used:

```sql
--Question 6

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

GO
```

Producing the following results:

100 %

Results | Messages

	supplier_name	product_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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 18 rows

..

100 %

Results | Messages

	supplier_name	product_name	reorder_level	quantity_in_stock
14	St. Jean's Company	Gorgonzola Telino	20	0
15	St. Jean's Company	Mascarpone Fabioli	25	9
16	Steveston Export Co...	Gravad lax	25	11
17	Steveston Export Co...	Outback Lager	30	15
18	Yves Delorme Ltd.	Longlife Tofu	5	4

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 18 rows

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.

Expected results:

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)

SQL statements used:

```sql
--Question 7

SELECT orders.order_id,
       customers.name,
        customers.contact_name,
        'shipped_date' = CONVERT(char(11), orders.shipped_date, 100),
        'elapsed' = CONVERT(char(8), DATEDIFF(year, orders.shipped_date, 'January 1, 2008 '))
FROM orders
INNER JOIN customers ON customers.customer_id = orders.customer_id
WHERE shipped_date is NOT NULL
ORDER BY order_id, elapsed
GO
```

Producing the following results:

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 1057 rows

--

100 %

Results | Messages

	order_id	name	contact_name	shipped_date	elapsed
1053	11063	Hungry Owl Al...	Patricia McKe...	Mar 30 2004	4
1054	11064	Save-a-lot Ma...	Jose Pavarotti	Mar 28 2004	4
1055	11066	White Clover ...	Karl Jablonski	Mar 28 2004	4
1056	11067	Drachenblut ...	Sven Ottlieb	Mar 30 2004	4
1057	11069	Tortuga Rest...	Miguel Angel ...	Mar 30 2004	4

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 1057 rows

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. The query should produce the result set listed below.

Expected results:

```
name        total
-----       -----------
A           4
B           7
C           5
D           3
E           2
...
T           6
V           3
W           5

(17 row(s) affected)
```

SQL statements used:

```sql
--Question 8
SELECT 'name' = LEFT(name, 1),
       'total' = COUNT(name)
FROM customers
GROUP BY LEFT(name, 1)
HAVING  COUNT(name) >=2 AND LEFT(name, 1) != 'S'
```

Producing the following results:

100 %

Results | Messages

	name	total
1	A	4
2	B	7
3	C	5
4	D	3
5	E	2

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 17 rows

--

100 %

Results | Messages

	name	total
13	Q	3
14	R	6
15	T	6
16	V	3
17	W	5

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 17 rows

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. The query should produce the result set listed below.

Expected results:

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)

SQL statements used:

```
--Question 9

SELECT order_details.order_id,
       order_details.quantity,
       products.product_id,
       products.reorder_level,
       products.supplier_id
FROM order_details
INNER JOIN products ON products.product_id = order_details.product_id
WHERE quantity > 100
ORDER BY order_id

GO
```

Producing the following results:

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 15 rows

--

100 %

Results | Messages

	order_id	quantity	product_id	reorder_level	supplier_id
11	10776	120	51	10	14
12	10894	120	75	25	12
13	10895	110	24	0	10
14	11017	110	59	0	8
15	11072	130	64	30	12

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 15 rows

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. The query should produce the result set listed below.

Expected results:

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)

SQL statements used:

```
--Question 10

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

GO
```

Producing the following results:

100 %

Results | Messages

	product_id	name	quantity_per_unit	unit_price
1	4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	22.00
2	5	Chef Anton's Gumbo Mix	36 boxes	21.35
3	74	Longlife Tofu	5 kg pkg.	10.00
4	14	Tofu	40 - 100 g pkgs.	23.25

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 4 rows

Part C - INSERT, UPDATE, DELETE and VIEWS Statements

Question 1:

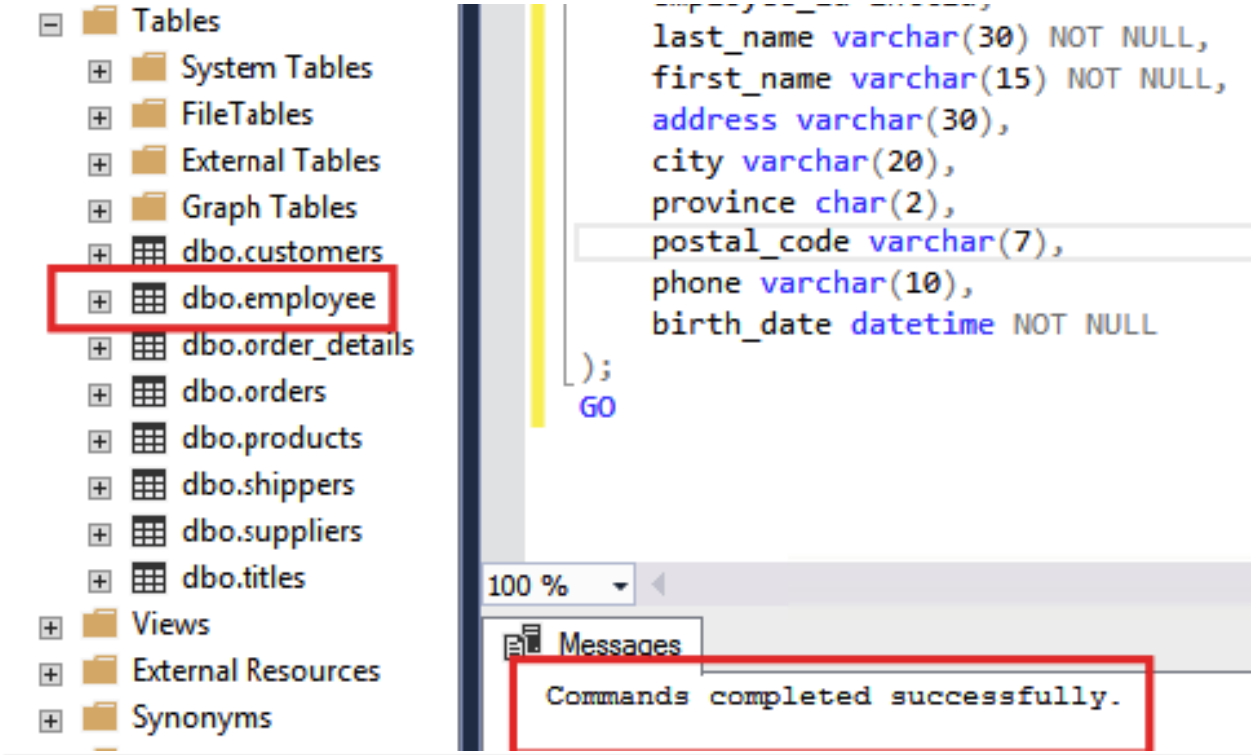
Create an **employee** table with the following columns:

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

SQL Statements used:

```sql
CREATE TABLE employee
(
    employee_id inttid,
    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
```

Producing the following result:



Question 2:

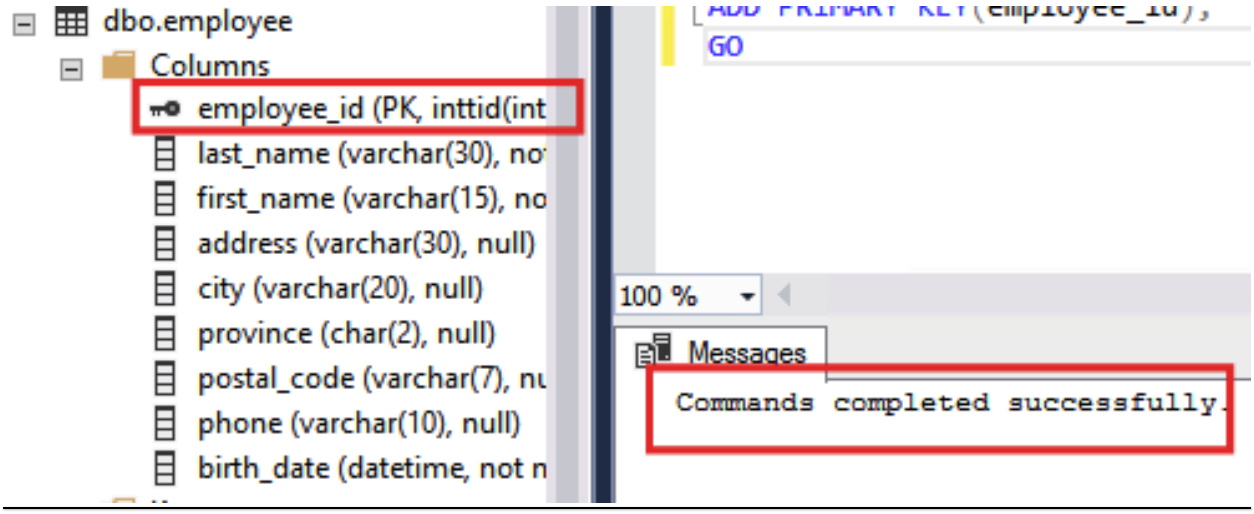
The **primary key** for the employee table should be the employee id.

SQL Statements used:

```
-- Question 2

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

Producing the following result:



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.

SQL Statements used:

```sql
-- Question 3

BULK INSERT employee
FROM 'C:\TextFiles\employee.txt'
WITH (          CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
      )
ALTER TABLE orders
ADD CONSTRAINT fk_orders_employee FOREIGN KEY(employee_id)
REFERENCES employee(employee_id);
GO
```

Producing the following results:

```
00 %
Messages

(9 rows affected)
```

Here we have created M:1 relationship between orders and employee entity.

```
dbo.orders
  Columns
    order_id (PK, inttid(int), no
    customer_id (FK, custtid(ch
    employee_id (FK, int, not n
    shipping_name (varchar(50
    shipping_address (varchar(
    shipping_city (varchar(20),
    shipping_region (varchar(1
    shipping_country_code (va

100 %
Messages
Commands completed successfully.
```

Question 4:

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

SQL Statements used:

```
--Question 4
INSERT INTO shippers(name)
VALUES('Quick Express')
GO
```

Producing the following results:

100 %

Messages

```
(1 row affected)
```

100 %

Results | Messages

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

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)

SQL Statements used:

```
-- Question 5
UPDATE products
SET unit_price = unit_price * 1.05
WHERE unit_price BETWEEN 5.00 AND 10.00
GO
```

Producing the following result:

```
100 %
Messages

(12 rows affected)
```

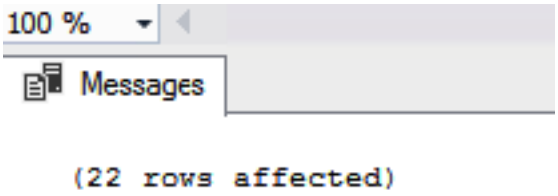
Question 6:

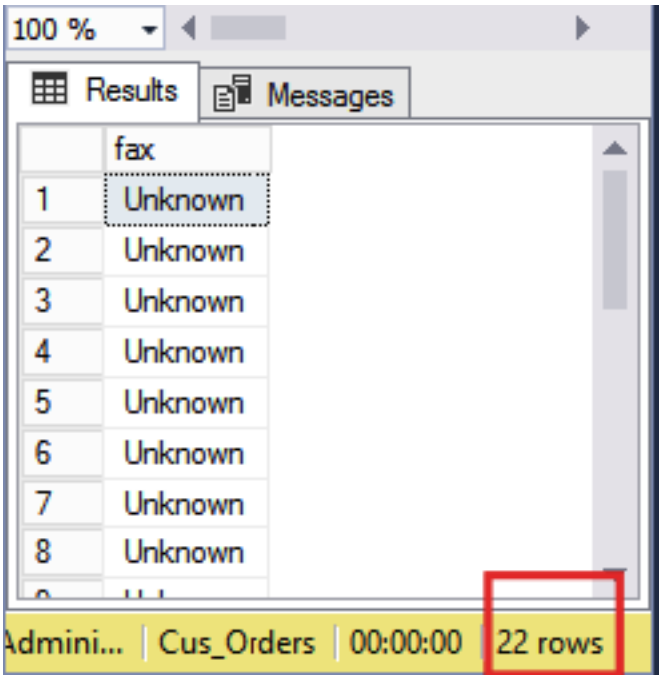
Using the UPDATE statement, change the fax value to **Unknown** for all rows in the customers table where the current fax value is **NULL**; (22 rows affected.)

SQL Statements used:

```
-- Question 6
UPDATE customers
SET fax = 'Unknown'
WHERE fax is NULL
GO
```

Producing the following 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 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**. The view should produce the result set listed below.

Expected results:

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)

SQL Statements used:

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

GO

SELECT *
FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200

GO
```

Producing the following results: (See challenge section)

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 540 rows

--

100 %

Results | Messages

	order_id	order_date	product_id	name	order_cost
534	10198	2002-03-26 00:00:00.000	46	Océano Atlántico Ltda.	72.00
535	10198	2002-03-26 00:00:00.000	56	Océano Atlántico Ltda.	684.00
536	10198	2002-03-26 00:00:00.000	76	Océano Atlántico Ltda.	540.00
537	10199	2002-03-27 00:00:00.000	1	Save-a-lot Markets	1188.00
538	10199	2002-03-27 00:00:00.000	3	Save-a-lot Markets	420.00
539	10199	2002-03-27 00:00:00.000	39	Save-a-lot Markets	720.00
540	10200	2002-03-30 00:00:00.000	11	Bólido Comidas preparadas	588.00

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 540 rows

---

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**. The view should produce the result set listed below.

Expected results:

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

SQL Statements used:

```
-- Question 8
CREATE VIEW vw_list_employees
AS
SELECT *
FROM employee
GO

SELECT employee_id,
       'name' = last_name + ', ' + first_name,
       'birth date' = CONVERT(char(11), birth_date, 102)
FROM vw_list_employees
WHERE employee_id IN (5, 7, 9)
GO
```

Producing the following results:

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 3 rows

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. The view should produce the result set listed below.

Expected results:

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)

SQL Statements used:

```sql
--Question 9
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 customers.customer_id = orders.customer_id
GO


SELECT order_id,
       customer_id,
       customer_name,
       city,
       country,
       'Shipped Date' = CONVERT(char(11), shipped_date, 100)
FROM vw_all_orders
WHERE shipped_date BETWEEN 'January 1, 2002' AND 'December 31, 2002'
ORDER BY customer_name, country
```

Producing the following results:

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 293 rows

--

100 %

Results | Messages

	order_id	customer_id	customer_name	city	country	Shipped Date
289	10420	WELLI	Wellington Importadora	Resende	Brazil	Dec 21 2002
290	10256	WELLI	Wellington Importadora	Resende	Brazil	Jun 10 2002
291	10269	WHITC	White Clover Markets	Seattle	United States	Jul 3 2002
292	10344	WHITC	White Clover Markets	Seattle	United States	Sep 29 2002
293	10374	WOLZA	Wolski Zajazd	Warszawa	Poland	Nov 2 2002

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 293 rows

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. The view should produce the result set listed below, *although not necessarily in the same order*.

Expected results:

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)

SQL Statements used:

```
CREATE VIEW vw_list_suppliers_items
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_list_suppliers_items
```

Producing the following results:

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 77 rows

--

100 %

Results | Messages

	supplier_id	supplier_name	product_id	product_name
73	7	Steveston Export Company	73	Röd Kaviar
74	4	Yves Delorme Ltd.	74	Longlife Tofu
75	12	South Harbour Products Ltd.	75	Rhönbräu Klosterbier
76	12	South Harbour Products Ltd.	76	Lakkalikööri
77	12	South Harbour Products Ltd.	77	Original Frankfurter grüne Soße

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 77 rows

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.

Expected results:

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

(6 row(s) affected)

SQL Statements used:

```sql
CREATE PROCEDURE sp_customer_city
(
   @CustomerCity varchar(20)
)
AS
   SELECT customer_id, name, address, city, phone
   FROM customers
   WHERE city = @CustomerCity
   GO

EXECUTE sp_customer_city 'London'
GO
```

Producing the following results:

100 %

Results | Messages

	customer_id	name	address	city	phone
1	AROUT	Around the Horn	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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 6 rows

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.

Expected results:

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)

SQL Statements used:

```sql
-- Question 2
CREATE PROCEDURE sp_orders_by_dates
(
    @starDate datetime,
    @endDate 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 customers.customer_id = orders.customer_id
INNER JOIN shippers ON shippers.shipper_id = orders.shipper_id
 WHERE orders.shipped_date BETWEEN @starDate AND @endDate
GO

EXECUTE sp_orders_by_dates 'January 1, 2003', 'June 30, 2003'
GO
```

Producing the following results:

100 %

Results | Messages

	order_id	customer_id	customer_name	shipper_name	shipped_date
1	10423	GOURL	Gourmet Lanchonetes	Federal Shipping	2003-01-18 00:00:00.000
2	10425	LAMAI	La maison d'Asie	United Package	2003-01-08 00:00:00.000
3	10427	PICCO	Piccolo und mehr	United Package	2003-01-25 00:00:00.000
4	10429	HUNGO	Hungry Owl All-Night Grocers	United Package	2003-01-01 00:00:00.000
5	10431	BOTTM	Bottom-Dollar Markets	United Package	2003-01-01 00:00:00.000

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 188 rows

---

100 %

Results | Messages

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 188 rows

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.

Expected results:

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)

SQL Statements used:

```sql
-- Question 3
CREATE PROCEDURE sp_product_listing
(
    @month varchar(10),
    @year int,
    @product varchar(50)
)
AS
SELECT 'product_name' = products.name,
       products.unit_price,
       products.quantity_in_stock,
       'supplier_name' = suppliers.name
FROM products
INNER JOIN suppliers ON suppliers.supplier_id = products.supplier_id
INNER JOIN order_details ON order_details.product_id = products.product_id
INNER JOIN orders ON order_details.order_id = orders.order_id
WHERE products.name LIKE '%' + @product + '%'
AND DATENAME(year, orders.order_date) = @year
AND  DATENAME(Month, orders.order_date)= @month
GO

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

Producing the following results:

100 %

Results | Messages

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 4 rows

Question 4:

Create a **DELETE** trigger on the order_details table to display the information shown below when you issue the following statement:

Expected 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

SQL Statements used:

```
-- Question 4

CREATE TRIGGER tr_delete_quantity
ON order_details
AFTER DELETE
AS
DECLARE @prod_id inttid, @qty_deletion int
SELECT  @prod_id = product_id, @qty_deletion = quantity
FROM deleted
UPDATE products
SET quantity_in_stock = quantity_in_stock + @qty_deletion
WHERE product_id = @prod_id
BEGIN
    SELECT 'Product_ID' = deleted.product_id,
           'Product NAme' = products.name,
           'Quantity being deleted from Order' = @qty_deletion,
           '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
```

Producing the following results:

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. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 1 rows

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.

SQL Statements used:

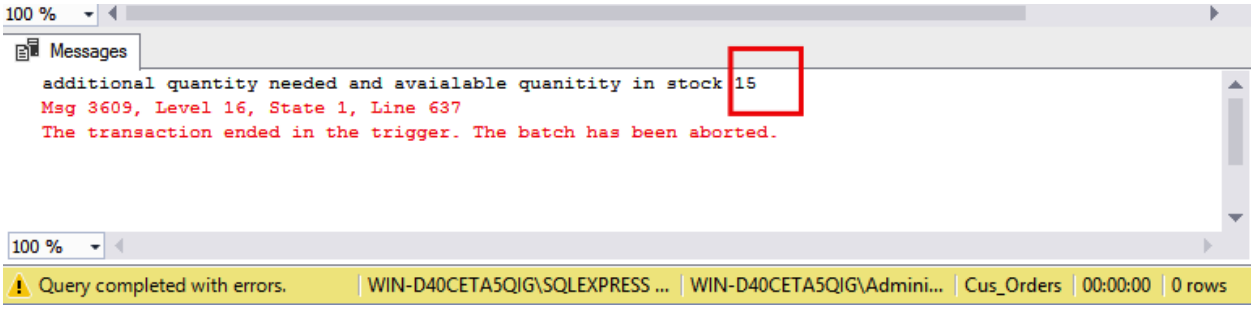
```sql
CREATE TRIGGER tr_qty_check
ON order_details
FOR UPDATE
AS
DECLARE @prod_id inttid, @gty int, @quantity_INSTOCK int
SELECT  @prod_id = products.product_id, @gty = inserted.quantity-deleted.quantity,
        @quantity_INSTOCK = products.quantity_in_stock
FROM inserted
INNER JOIN deleted ON inserted.product_id = deleted.product_id
INNER JOIN products ON inserted.product_id = products.product_id
IF(@gty > @quantity_INSTOCK)
BEGIN
     PRINT 'additional quantity needed and avaialable quanitity in stock '
           + CONVERT(char(11), @quantity_INSTOCK)
     ROLLBACK TRANSACTION
END
ELSE
BEGIN
   UPDATE products
   SET quantity_in_stock = quantity_in_stock - @gty
   WHERE product_id = @prod_id
   --Display updated quantity
   SELECT 'updated quantity in stock' = products.quantity_in_stock
   FROM products
   WHERE products.product_id = @prod_id
END
GO
```

Question 6:

Run the following 2 queries separately to verify your trigger:

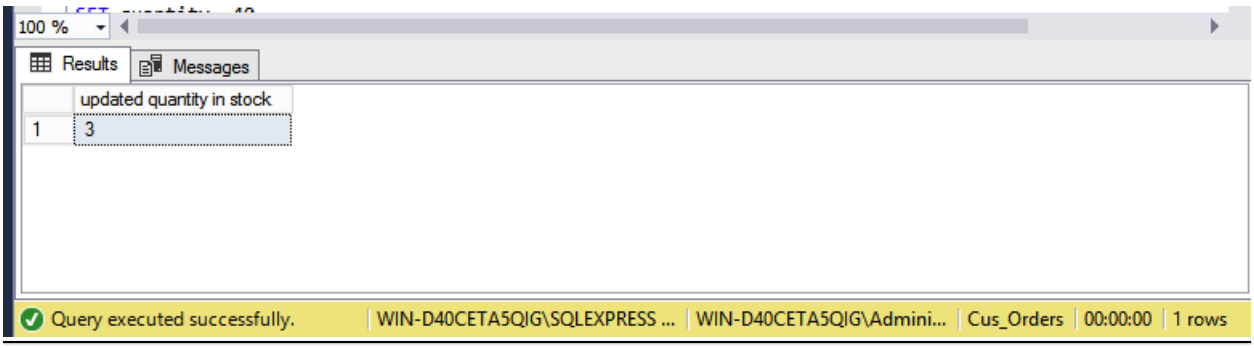
- UPDATE order_details
  SET quantity =50
  WHERE order_id = '10044'
  AND product_id = 7;

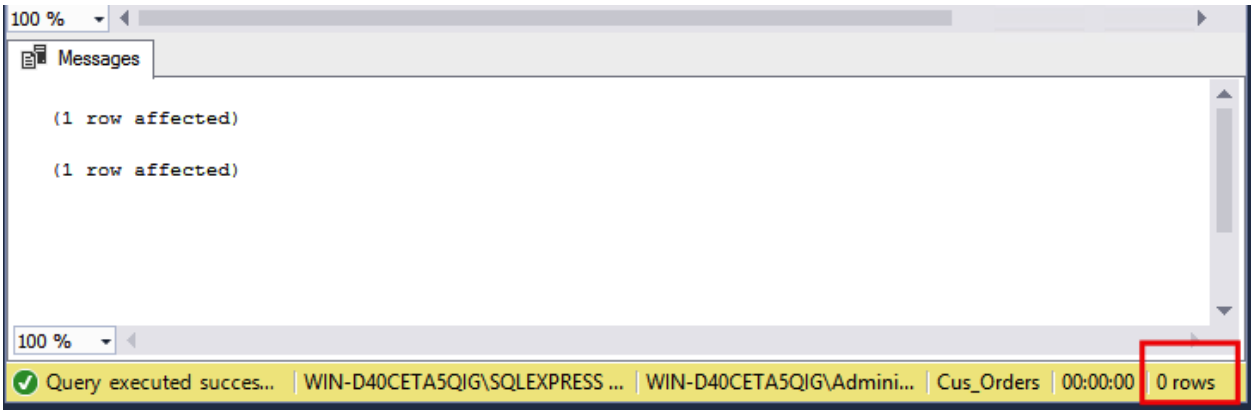
Producing the following results:



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

Producing the following results:





Question 7:

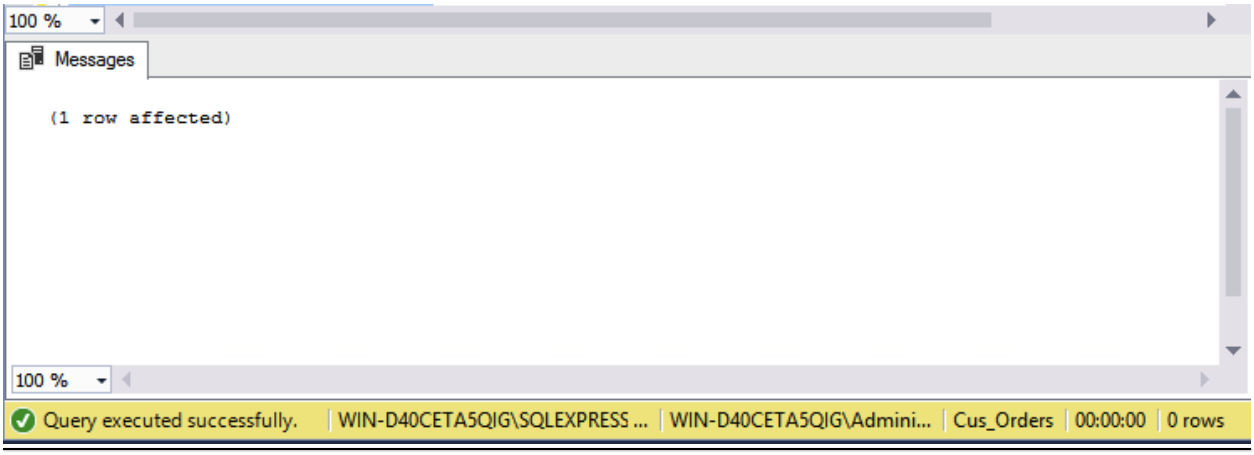
Create a stored procedure called **sp_del_inactive_cust** to **delete** customers that have no orders. The stored procedure should delete **1** row.

SQL Statements used:

```
CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE
FROM customers
WHERE customers.customer_id NOT IN
    (SELECT orders.customer_id
    FROM orders)
GO

EXECUTE sp_del_inactive_cust
GO
```

Producing the following 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.

Expected results:

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

(1 row(s) affected)

SQL Statements used:

```
--Question 8
CREATE PROCEDURE sp_employee_information
(
    @emp_id int
)
AS
SELECT *
FROM employee
WHERE employee_id = @emp_id
GO

EXECUTE sp_employee_information 5
GO
```

Producing the following results

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 1 rows

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.

Expected results:

	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 Co...	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 We...	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 We...	BC	4	5

(24 row(s) affected)

SQL Statements used:

```sql
CREATE PROCEDURE sp_reorder_qty
(
    @unit int
)
AS
SELECT products.product_id,
        suppliers.name,
        suppliers.address,
        suppliers.city,
        suppliers.province,
        'qty' = products.quantity_in_stock,
        products.reorder_level
FROM products
INNER JOIN suppliers ON suppliers.supplier_id = products.supplier_id
WHERE (quantity_in_stock - reorder_level ) < @unit
GO

EXECUTE sp_reorder_qty 5
GO
```

Producing the following results:

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 24 rows

---

100 %

Results | Messages

	product_id	name	address	city	province	qty	reorder_level
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 We	BC	4	5

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 24 rows

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.

Expected results:

product_id	name	alternate_name	unit_price
13	Konbu	Kelp Seaweed	6.30
19	Teatime Chocolate Biscuits	Teatime Chocolate Biscuits	9.66
23	Tunnbr÷d	Thin Bread	9.45
45	R°gede sild	Smoked Herring	9.975
47	Zaanse koeken	Zaanse Cookies	9.975
52	Filo Mix	Mix for Greek Filo Dough	7.35
54	TourtiÞre	Pork Pie	7.8225
75	Rh÷nbrõu Klosterbier	Rh÷nbrõu Beer	8.1375

(8 row(s) affected)

SQL Statements used:

```sql
CREATE PROCEDURE sp_unit_prices
(
    @unit_price1 int,
    @unit_price2 int
)
AS
SELECT products.product_id,
       products.name,
       products.alternate_name,
       unit_price
FROM products
WHERE unit_price BETWEEN @unit_price1 AND @unit_price2
GO

EXECUTE sp_unit_prices $5.00, $10.00
GO
```

Producing the following results:

100 %

Results | Messages

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

Query executed successfully. | WIN-D40CETA5QIG\SQLEXPRESS ... | WIN-D40CETA5QIG\Admini... | Cus_Orders | 00:00:00 | 8 rows

3. CHALLENGES AND CONCLUSION

All desired results have been achieved, and there is no challenge. However, question 5 was a tough question than other questions. In this question, you need to know what happens when you update the two tables, and you need to understand that where old value and new inserted value are stored when you update any column. Therefore, inserted and deleted table have been taken into consideration in this type of question.

As conclusion, I have learnt a lot from this project. Especially, I have enjoyed in part D. This project has been a great learning experience.

4.SQL SCRIPTS AND DB DIAGRAM

```
/*----------- EMRE PACACI A01029808-----------------------*/
/*----------------PART A ---------------------------------*/

/*-------------- A-Question 1 ----------------------------*/

/* Make sure the Master database (the overall container da-
tabase ) is selected */
USE master;
GO

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

CREATE DATABASE Cus_Orders;
GO

USE Cus_Orders;
GO

/*-------------- A-Question 2 ----------------------------*/

CREATE TYPE custtid FROM char(5) NOT NULL;
CREATE TYPE inttid FROM int NOT NULL;
GO
```

```sql
/*------------- A-Question 3 ------------------------------*/

 /* Create Customer table */
 CREATE TABLE customers
 (
     customer_id custtid,
     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 orders table*/
 CREATE TABLE orders
 (
     order_id inttid,
     customer_id custtid,
     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
 );
```

```sql
/* create order_details table*/
 CREATE TABLE order_details
 (
     order_id inttid,
     product_id inttid,
     quantity int NOT NULL,
     discount float NOT NULL
 );
 /* Create products table */
 CREATE TABLE products
 (
     product_id inttid,
     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 shippers table*/
 CREATE TABLE shippers
 (
    shipper_id int IDENTITY NOT NULL,
     name varchar(20)
 );

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

```sql
/* Create titles table */
CREATE TABLE titles
(
    title_id char(3) NOT NULL,
    description varchar(35)
);

GO

/*-------- A-Question 4 -------------------------------*/

ALTER TABLE customers
ADD PRIMARY KEY (customer_id);

ALTER TABLE orders
ADD PRIMARY KEY (order_id);

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

ALTER TABLE products
ADD PRIMARY KEY (product_id);

ALTER TABLE shippers
ADD PRIMARY KEY (shipper_id);

ALTER TABLE suppliers
ADD PRIMARY KEY (supplier_id);

ALTER TABLE titles
ADD PRIMARY KEY (title_id);

GO

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

```sql
ALTER TABLE orders
ADD CONSTRAINT fk_order_cus FOREIGN KEY(customer_id)
REFERENCES customers(customer_id);

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

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

ALTER TABLE order_details
ADD CONSTRAINT fk_order_details_product FOREIGN KEY (prod-
uct_id)
REFERENCES products (product_id);

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

/*--------- A-Question 5 ------------------------------*/

ALTER TABLE customers
ADD CONSTRAINT default_Country DEFAULT('Canada') FOR coun-
try;

ALTER TABLE orders
ADD CONSTRAINT default_date DEFAULT(GETDATE() + 10) FOR re-
quired_date;

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

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

```sql
ALTER TABLE products
ADD CONSTRAINT check_quantity_in_stoct CHECK (quan-
tity_in_stock < 150);

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

GO

/*--------- A-Question 6 ----------------------------*/

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'
       )
```

```sql
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'
       )

GO

/*----------------PART B----------------------------------*/

/*------------ B-Question 1-------------------------------*/

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

/*------------ B-Question 2-------------------------------*/

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

```
/*------------ B-Question 3-------------------------------*/

SELECT  orders.order_id,
        'product_name' = products.name,
        'customer_name' = customers.name,
        'order_date'= CONVERT(char(11), orders.order_date,
100),
        'new_shipped_date' = CONVERT(char(11), DATEADD(day,
17,orders.shipped_date), 100),
        'order_cost' = (order_details.quantity * prod-
ucts.unit_price)
FROM orders
INNER JOIN customers ON orders.customer_id = customers.cus-
tomer_id
INNER JOIN order_details ON order_details.order_id = or-
ders.order_id
INNER JOIN products ON products.product_id = order_de-
tails.product_id
WHERE order_date BETWEEN 'January 1, 2001' AND 'December 31,
2001'
GO

/*------------ B-Question 4-------------------------------*/

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

```sql
/*------------ B-Question 5-------------------------------*/

SELECT customers.customer_id,
       customers.name,
        customers.city,
        titles.description
FROM customers
INNER JOIN titles ON customers.title_id = titles.title_id
WHERE customers.region is NULL
GO

/*------------ B-Question 6-------------------------------*/

SELECT 'supplier_name' = suppliers.name,
       'product_name' = products.name,
        products.reorder_level,
        products.quantity_in_stock
FROM suppliers
INNER JOIN products ON products.supplier_id = suppliers.sup-
plier_id
WHERE reorder_level > quantity_in_stock
ORDER BY supplier_name
GO

/*------------ B-Question 7-------------------------------*/

SELECT orders.order_id,
       customers.name,
        customers.contact_name,
       'shipped_date' =
        CONVERT(char(11), orders.shipped_date, 100),
       'elapsed' = CONVERT(char(8), DATEDIFF(year,
                   orders.shipped_date, 'January 1, 2008 '))
FROM orders
INNER JOIN customers ON customers.customer_id = orders.cus-
tomer_id
WHERE shipped_date is NOT NULL
ORDER BY order_id, elapsed
```

```sql
GO

/*------------ B-Question 8-------------------------------*/

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

/*------------ B-Question 9-------------------------------*/

SELECT order_details.order_id,
       order_details.quantity,
        products.product_id,
        products.reorder_level,
        products.supplier_id
FROM order_details
INNER JOIN products ON products.product_id = order_de-
tails.product_id
WHERE quantity > 100
ORDER BY order_id
GO

/*------------ B-Question 10------------------------------*/

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

```sql
/*--------------PART C-----------------------------------*/

/*------- C-Question 1 -----------------------------------*/

CREATE TABLE employee
(
     employee_id inttid,
     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

/*------- C-Question 2-------------------------------*/

ALTER TABLE employee
ADD PRIMARY KEY(employee_id);
GO

/*------- C-Question 3 -------------------------------*/

BULK INSERT employee
FROM 'C:\TextFiles\employee.txt'
WITH (        CODEPAGE=1252,
          DATAFILETYPE = 'char',
          FIELDTERMINATOR = '\t',
          KEEPNULLS,
          ROWTERMINATOR = '\n'
      )
ALTER TABLE orders
ADD CONSTRAINT fk_orders_employee FOREIGN KEY(employee_id)
REFERENCES employee(employee_id);
```

```sql
GO
/*------- C-Question 4 -------------------------------*/

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

/*------- C-Question 5 -------------------------------*/

UPDATE products
SET unit_price = unit_price * 1.05
WHERE unit_price BETWEEN 5.00 AND 10.00
GO

/*------- C-Question 6 -------------------------------*/

UPDATE customers
SET fax = 'Unknown'
WHERE fax is NULL
GO

/*------- C-Question 7 -------------------------------*/

CREATE VIEW vw_order_cost
AS
SELECT orders.order_id,
       orders.order_date,
        products.product_id,
        customers.name,
        'order_cost' = (order_details.quantity * prod-
ucts.unit_price)
FROM orders
INNER JOIN order_details ON order_details.order_id = or-
ders.order_id
INNER JOIN products ON order_details.product_id = prod-
ucts.product_id
INNER JOIN customers ON customers.customer_id = orders.cus-
tomer_id
```

```sql
GO

SELECT *
FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200
GO

/*------- C-Question 8 -----------------------------------*/

CREATE VIEW vw_list_employees
AS
SELECT *
FROM employee
GO

SELECT employee_id,
       'name' = last_name + ', ' + first_name,
        'birth_date' = CONVERT(char (11), birth_date, 102)
FROM vw_list_employees
WHERE employee_id IN (5, 7, 9)
GO

/*------- C-Question 9 ----------------------------------*/

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 customers.customer_id = orders.cus-
tomer_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 'January 1, 2002' AND 'December
31, 2002'
ORDER BY customer_name, country
GO

/*------- C-Question 10 -------------------------------*/

CREATE VIEW vw_list_suppliers_items
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.sup-
plier_id
GO

SELECT *
FROM vw_list_suppliers_items
GO
```

```sql
/*------------------PART D -------------------------------*/

/*-------------- D-Question 1 ----------------------------*/

CREATE PROCEDURE sp_customer_city
(
   @CustomerCity varchar(20)
)
AS
   SELECT customer_id, name, address, city, phone
   FROM customers
   WHERE city = @CustomerCity
   GO

EXECUTE sp_customer_city 'London'
GO

/*-------------- D-Question 2 ----------------------------*/

CREATE PROCEDURE sp_orders_by_dates
(
     @starDate datetime,
     @endDate 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 customers.customer_id = orders.cus-
tomer_id
INNER JOIN shippers ON shippers.shipper_id = orders.ship-
per_id
WHERE orders.shipped_date BETWEEN @starDate AND @endDate
GO
```

```
EXECUTE sp_orders_by_dates 'January 1, 2003', 'June 30,
2003'
GO

/*---------------- D-Question 3 --------------------------*/

CREATE PROCEDURE sp_product_listing
(
     @month varchar (10),
     @year int,
     @product varchar (50)
)
AS
SELECT 'product_name' = products.name,
        products.unit_price,
        products.quantity_in_stock,
        'supplier_name' = suppliers.name
FROM products
INNER JOIN suppliers ON suppliers.supplier_id =
                         products.supplier_id
INNER JOIN order_details ON order_details.product_id =
                            products.product_id
INNER JOIN orders ON order_details.order_id = orders.or-
der_id
WHERE products.name LIKE '%' + @product + '%'
AND DATENAME(year, orders.order_date) = @year
AND  DATENAME(Month, orders.order_date)= @month
GO

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

```sql
/*------------ D-Question 4 -------------------------------*/

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

GO

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

```sql
/*---------------- D-Question 5----------------------*/

CREATE TRIGGER tr_qty_check
ON order_details
FOR UPDATE
AS
DECLARE @prod_id inttid, @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.prod-
uct_id
INNER JOIN products ON inserted.product_id = products.prod-
uct_id
IF (@gty > @quantity_INSTOCK)
BEGIN
     PRINT 'additional quantity needed and available quan
     itity in stock '
            + CONVERT(char(11), @quantity_INSTOCK)
     ROLLBACK TRANSACTION
END
ELSE
BEGIN
   UPDATE products
   SET quantity_in_stock = quantity_in_stock - @gty
   WHERE product_id = @prod_id
   --Display updated quantity
   SELECT 'updated quantity in stock' =
           products.quantity_in_stock
   FROM products
   WHERE products.product_id = @prod_id
END
GO
```

```sql
/*----------------- D-Question 6 -------------------------*/

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;
GO

/*---------------- D-Question 7 --------------------------*/

CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE
FROM customers
WHERE customers.customer_id NOT IN
     (SELECT orders.customer_id
     FROM orders)
GO

EXECUTE sp_del_inactive_cust
GO
/*------------------ D-Question 8 -----------------------*/

CREATE PROCEDURE sp_employee_information
(
     @emp_id int
)
AS
SELECT *
FROM employee
WHERE employee_id = @emp_id
GO

EXECUTE sp_employee_information 5
```

```sql
GO
/*---------------- D-Question 9 --------------------------*/

CREATE PROCEDURE sp_reorder_qty
(
    @unit int
)
AS
SELECT products.product_id,
         suppliers.name,
         suppliers.address,
         suppliers.city,
         suppliers.province,
         'qty' = products.quantity_in_stock,
         products.reorder_level
FROM products
INNER JOIN suppliers ON suppliers.supplier_id = prod-
ucts.supplier_id
WHERE (quantity_in_stock - reorder_level ) < @unit
GO

EXECUTE sp_reorder_qty 5
GO

/*---------------- D-Question 10 --------------------------*/

CREATE PROCEDURE sp_unit_prices
(
    @unit_price1 int,
    @unit_price2 int
)
AS
SELECT products.product_id,
       products.name,
       products.alternate_name,
       unit_price
FROM products
WHERE unit_price BETWEEN @unit_price1 AND @unit_price2
```

```sql
GO
EXECUTE sp_unit_prices $5.00, $10.00
GO
```

Database Diagram: