

COMP 1630 Project 2 - SQL queries

Jae Sung Kim

Table of Contents

Φ Checking for existing DB	4
 1 Part A Database and Tables	
1.1 Part A Question 1	5
1.2 Part A Question 2	5
1.3 Part A Question 3	6
1.4 Part A Question 4	8
1.5 Part A Question 5	10
1.6 Part A Question 6	11
 2 Part B SQL Statements	
2.1 Part B Question 1	14
2.2 Part B Question 2	16
2.3 Part B Question 3	17
2.4 Part B Question 4	19
2.5 Part B Question 5	21
2.6 Part B Question 6	23
2.7 Part B Question 7	25
2.8 Part B Question 8	27
2.9 Part B Question 9	29
2.10 Part B Question 10	31
 3 Part C INSERT, UPDATE, DELETE and VIEWS Statements	
3.1 Part C Question 1	33
3.2 Part C Question 2	34
3.3 Part C Question 3	34
3.4 Part C Question 4	35
3.5 Part C Question 5	36
3.6 Part C Question 6	37

3.7 Part C Question 7	38
3.8 Part C Question 8	40
3.9 Part C Question 9	42
3.10 Part C Question 10	44

4 Part D Stored Procedures and Triggers

4.1 Part D Question 1	46
4.2 Part D Question 2	48
4.3 Part D Question 3	50
4.4 Part D Question 4	52
4.5 Part D Question 5	53
4.6 Part D Question 6	54
4.7 Part D Question 7	55
4.8 Part D Question 8	56
4.9 Part D Question 9	58

4 Part D Stored Procedures and Triggers

5.1 Query File	60
--------------------------	----

1 Part A - Database and Tables

Φ Checking for existing DB

Query

```
USE master;                                --select master which has all the databases

GO

IF EXISTS                                  --checking for the existence of 'Cus_Orders' database
                                           --deletes it if found
(
    SELECT *
    FROM sysdatabases
    WHERE name='Cus_Orders'
)
BEGIN
    PRINT('Dropping database...')
    DROP DATABASE Cus_Orders;
END
GO
```

Results



Notes and Observations

- Deletion of databases requires that the focus is on “master” or else the operation fails

1 Part A - Database and Tables

1.1 Part A Question 1

Create a database called Cus_Orders.

Query

```
USE master;  
GO  
  
/*#1*/  
  
CREATE DATABASE Cus_Orders;  
GO  
  
USE Cus_Orders;  
GO
```

Results



1.2 Part A 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. See pages 12/13 for specifications.

Query

```
CREATE TYPE uniqid FROM CHAR(5) NOT NULL;  
CREATE TYPE intid FROM int NOT NULL;  
GO
```

1 Part A - Database and Tables

1.3 Part A Question 3

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

customers

orders

order_details

products

shippers

suppliers

titles

Query (Part 1)

```
CREATE TABLE customers
```

```
(
    customer_id uniqueid,
    name VARCHAR(50),
    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 uniqueid,
    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
);
```

1 Part A - Database and Tables

Query (Part 2)

```
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),
    alternate_name VARCHAR(40),
    quantity_per_unit VARCHAR(25),
    unit_price money,
    quantity_in_stock int,
    units_on_stock int,
    reorder_level int
);
```

```
CREATE TABLE shippers
(
    shipper_id int IDENTITY(1,1) NOT NULL,
    name VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE suppliers
(
    supplier_id int IDENTITY(1,1) 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
```

1 Part A - Database and Tables

1.4 Part A Question 4

Set the primary keys and foreign keys for the tables.

Query (Setting Pks)

```
ALTER TABLE customers
```

```
ADD PRIMARY KEY (customer_id);
```

```
GO
```

```
ALTER TABLE orders
```

```
ADD PRIMARY KEY (order_id);
```

```
GO
```

```
ALTER TABLE order_details
```

```
ADD PRIMARY KEY (order_id, product_id);
```

```
GO
```

```
ALTER TABLE products
```

```
ADD PRIMARY KEY (product_id);
```

```
GO
```

```
ALTER TABLE shippers
```

```
ADD PRIMARY KEY (shipper_id);
```

```
GO
```

```
ALTER TABLE suppliers
```

```
ADD PRIMARY KEY (supplier_id);
```

```
GO
```

```
ALTER TABLE titles
```

```
ADD PRIMARY KEY (title_id);
```

```
GO
```


1 Part A - Database and Tables

Query (Setting Fks)

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

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

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

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

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

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

1 Part A - Database and Tables

1.5 Part A Question 5

Set the constraints as follows:

Table Name	Constraint
Customer	country should default to Canada
Orders	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

Query

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

ALTER TABLE orders
ADD CONSTRAINT default_date
    DEFAULT(DATEADD(DAY,10,GETDATE())) FOR order_date;
GO

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

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

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

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

1 Part A - Database and Tables

1.6 Part A Question 6

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

File name	Allocation	Number of rows
Customers.txt	Into the customers table	91
Orders.txt	Into the orders table	1078
order_details.txt	Into the order_details table	2820
Products.txt	Into the products table	77
Shippers	Into the shippers table	3
Titles	Into the titles	15

Query (Data load script part 1)

```
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'
)
```

1 Part A - Database and Tables

Query (Data load script part 2)

```
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'
)
```

1 Part A - Database and Tables

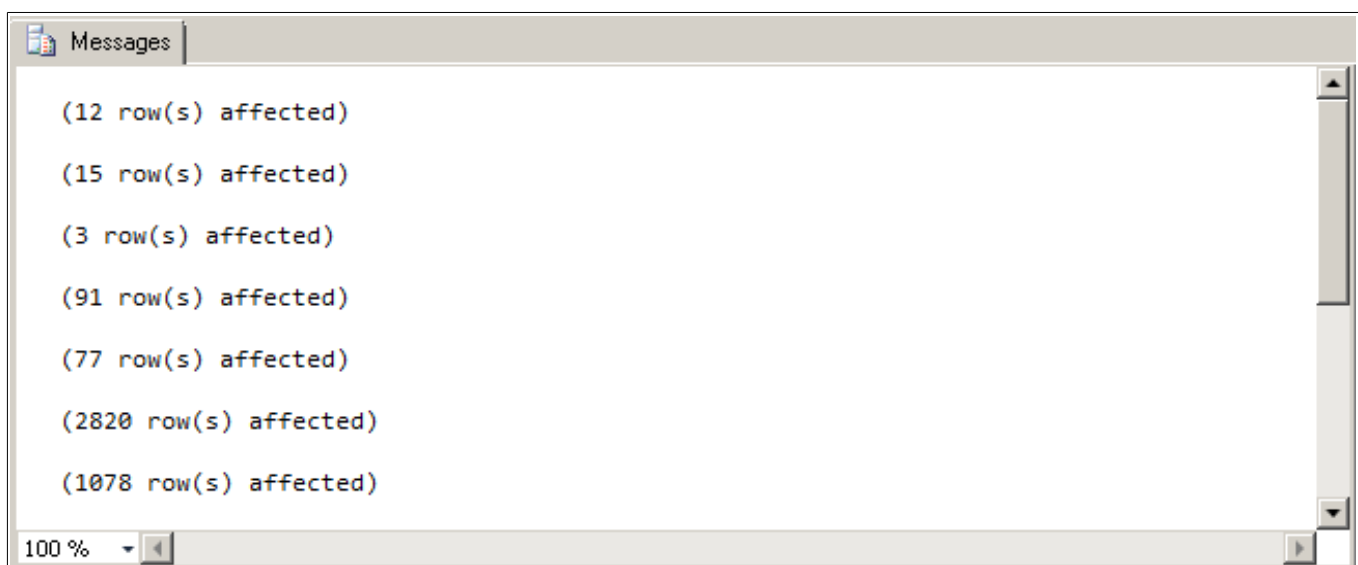
Query (Data load script part 3)

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

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

GO
```

Results



2 Part B - SQL Statements

2.1 Part B 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.

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)			

Query

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

Notes and Observations

- Ordering by 'customer_id' is determined by the first letter, then the second letter, and so on.

2 Part B - SQL Statements

Results (rows 1 to 13 from 91rows)

	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

	customer_id	name	city	country
(91 row(s) affected)				

2 Part B - SQL Statements

2.2 Part B 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.

Query

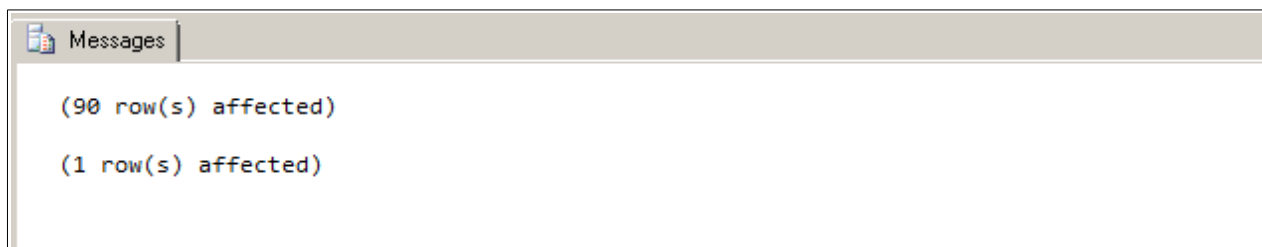
```
ALTER TABLE customers
ADD active bit;
GO

ALTER TABLE customers
ADD CONSTRAINT default_active
    DEFAULT (1) FOR active;
GO

UPDATE customers
SET active=1
WHERE EXISTS
(
    SELECT *
    FROM orders
    WHERE orders.customer_id=customers.customer_id
)
GO

UPDATE customers
SET active=0
WHERE NOT EXISTS
(
    SELECT *
    FROM orders
    WHERE orders.customer_id=customers.customer_id
)
GO
```

Results



Notes and Observations

- customers with orders and those without orders were selected with appropriate queries and their 'active' columns were updated with either a '0' or a '1' value.

2 Part B - SQL Statements

2.3 Part B 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 7 days to the shipped date from the orders table, the product name from the product table, the customer name from the customer table, and the cost of the order. Format the date order date and the shipped date as MON DD YYYY. Use the formula (quantity * unit_price) to calculate the cost of the order. The query should produce the result set listed below.

order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
10000	Alice Mutton	Franchi S.p.A.	May 10 2001	May 22 2001	156.0000
10001	NuNuCa Nuß-Nougat-Crème Mère	Paillarde May	May 13 2001	May 30 2001	420.0000
10001	Boston Crab Meat	Mère Paillarde	May 13 2001	May 30 2001	736.0000
10001	Raclette Courdavault	Mère Paillarde	May 13 2001	May 30 2001	440.0000
10001	Wimmers gute Semmelknödel	Mère Paillarde	May 13 2001	May 30 2001	498.7500
10138	Inlagd Sill	Du monde entire	Dec 27 2001	Jan 10 2002	228.0000
10138	Louisiana Hot Spiced Okra	Du monde entire	Dec 27 2001	Jan 10 2002	204.0000
10139	Camembert Pierrot	Vaffeljernet	Dec 30 2001	Jan 16 2002	680.0000
(383 row(s) affected)					

2 Part B - SQL Statements

Query

```
SELECT 'order_id'=orders.order_id,  
       'product_name'=products.name,  
       'customer_name'=customers.name,  
       'order_date'= CONVERT(CHAR(11),orders.order_date,0),  
       'new_shipped_date'= CONVERT(CHAR(11),DATEADD(DAY,7,orders.order_date)),  
       'order_cost'= order_details.quantity*products.unit_price  
FROM order_details  
INNER JOIN orders 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  
WHERE orders.order_date BETWEEN 'January 1, 2001' AND 'December 31, 2001';
```

Results (rows 1 to 13 from 383 rows)

	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
1	10000	Alice Mutton	Franchi S.p.A.	May 10 2001	May 17 2001	156.00
2	10001	NuNuCa Nuß-Nougat-Creme	Mère Paillarde	May 13 2001	May 20 2001	420.00
3	10001	Boston Crab Meat	Mère Paillarde	May 13 2001	May 20 2001	736.00
4	10001	Raclette Courdavault	Mère Paillarde	May 13 2001	May 20 2001	440.00
5	10001	Wimmers gute Semmelknödel	Mère Paillarde	May 13 2001	May 20 2001	498.75
6	10002	Gorgonzola Telino	Folk och få HB	May 14 2001	May 21 2001	437.50
7	10002	Chartreuse verte	Folk och få HB	May 14 2001	May 21 2001	324.00
8	10002	Fløtemysost	Folk och få HB	May 14 2001	May 21 2001	322.50
9	10003	Camarvon Tigers	Simons bistro	May 15 2001	May 22 2001	750.00
10	10004	Thüringer Rostbratwurst	Vaffeljernet	May 16 2001	May 23 2001	4332.65
11	10004	Veggie-spread	Vaffeljernet	May 16 2001	May 23 2001	263.40
12	10005	Tarte au sucre	Wartian Herkku	May 20 2001	May 27 2001	295.80
13	10006	Konbu	Franchi S.p.A.	May 21 2001	May 28 2001	60.00

Notes and Observations

Results	Messages
(383 row(s) affected)	

- Three inner joins are required to connect the tables order_details, orders, products and customers.

2 Part B - SQL Statements

2.4 Part B 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. The query should produce the result set listed below. Your displayed results may look slightly different to those shown below but the query should still return 21 rows.

customer_id	Name	Phone	order_id	order_date
BLAUS	Blauer See Delikatessen	0621-08460	11058	2004-03-23 00:00:00.000
BONAP	Bon app'	91.24.45.40	11076	2004-03-30 00:00:00.000
ERNSH	Ernst Handel	7675-3425	11008	2004-03-02 00:00:00.000
.....				
RICAR	Ricardo Adocicados	(21) 555-3412	11059	2004-03-23 00:00:00.000
RICSU	Richter Supermarkt	0897-034214	11075	2004-03-30 00:00:00.000
SIMOB	Simons bistro	31 12 34 56	11074	2004-03-30 00:00:00.000
(21 row(s) affected)				

Query

```
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;
```

Notes and Observations

- “..have not been shipped” means records with a shipped_date field set to NULL, hence the query above.

2 Part B - SQL Statements

Results (rows 1 to 13 from 21 rows)

	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	LIND-Delicatesses	(8) 34-56-12	11039	2004-03-15 00:00:00.000

Results	Messages
(21 row(s) affected)	

2 Part B - SQL Statements

2.5 Part B 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. 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)			

Query

```
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;
```

2 Part B - SQL Statements

Results (1 to 13 of 60 rows)

	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

Results Messages

```
(60 row(s) affected)
|
```

2 Part B - SQL Statements

2.6 Part B 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. The query should produce the result set listed below.

supplier_name	product_name	reorder_level	quantity_in_stock
Armstrong Company	Queso Cabrales	30	22
Cadbury Products Ltd.	Ipoh Coffee	25	17
Cadbury Products Ltd.	Røgede sild	15	5
Campbell Company	Gnocchi di nonna Alice	30	21
Dare Manufacturer Ltd.	Gnocchi di nonna Alice	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)			

Query

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

2 Part B - SQL Statements

Results (rows 1 to 13 from 18 rows)

	name	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

Results	Messages
<pre>(18 row(s) affected)</pre>	

2 Part B - SQL Statements

2.7 Part B 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. 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 få 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)

Query

```
SELECT orders.order_id,
       customers.name,
       customers.contact_name,
       'shipped_date'=CONVERT(CHAR(11),orders.shipped_date, 100),
       'elapsed'=DATEDIFF(YEAR, orders.shipped_date, 'January 1, 2008')
FROM orders
INNER JOIN customers ON orders.customer_id=customers.customer_id
WHERE orders.shipped_date IS NOT NULL
ORDER BY orders.order_id, 'elapsed';
```

2 Part B - SQL Statements

Results (rows 1 to 13 from 1075 rows)

	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 få HB	Maria Larsson	May 17 2001	7
4	10003	Simons bistro	Jytte Petersen	May 24 2001	7
5	10004	Vaifeljernet	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	LIND-Delicateses	Felipe Izquierdo	Jun 3 2001	7

Results	Messages
(1057 row(s) affected)	

2 Part B - SQL Statements

2.8 Part B 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.

name	total
A	4
B	7
C	5
D	3
E	2
...	
T	6
V	3
W	5
(17 row(s) affected)	

Query

```
SELECT 'name' = SUBSTRING(name,1,1),  
       'total' = COUNT(name)  
FROM customers  
WHERE SUBSTRING(name,1,1) LIKE '[^S]'  
GROUP BY SUBSTRING(name,1,1)  
HAVING COUNT(name) >= 2;
```

2 Part B - SQL Statements

Results (rows 1 to 13 from 17 rows)

	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

Results	Messages
(17 row(s) affected)	

2 Part B - SQL Statements

2.9 Part B 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 and reorder level from the products table, and the supplier id from the suppliers table. Order the result set by the order id. The query should produce the result set listed below.

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

Query

```
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 suppliers.supplier_id=products.supplier_id  
WHERE order_details.quantity > 100  
ORDER BY order_details.order_id;
```

2 Part B - SQL Statements

Results

	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

Results	Messages
(15 row(s) affected)	

2 Part B - SQL Statements

2.10 Part B 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.

product_id	Name	quantity_per_unit	unit_price
4	Chef Anton's Cajon Seasoning	48 – 6 oz jars	22.0000
5	Chef Anton's Gumbo Mix	36 boxes	21.3500
74	Longlife Tofu	5kg pkg.	10.0000
14	Tofu	40-100g pkgs.	23.2500
(4 row(s) affected)			

Query

```
SELECT products.product_id,  
       products.name,  
       products.quantity_per_unit,  
       products.unit_price  
FROM products  
WHERE products.name LIKE '%tofu%'  
UNION ALL  
SELECT products.product_id,  
       products.name,  
       products.quantity_per_unit,  
       products.unit_price  
FROM products  
WHERE products.name LIKE '%chef%'  
ORDER BY products.name;
```

Notes and Observations

- '%tofu%' and '%chef%' ensures matches anywhere within the field value.

2 Part B - SQL Statements

Results

	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

	product_id	name	quantity_per_unit	unit_price
(4 row(s) affected)				

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

3.1 Part C 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

Query

```
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
);
GO
```

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

3.2 Part C Question 2

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

Query

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

3.3 Part C 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.

Query

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

ALTER TABLE orders
ADD CONSTRAINT FK_employee_orders FOREIGN KEY(employee_id)
REFERENCES employee(employee_id);
GO
```

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

3.4 Part C Question 4

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

Query

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

```
SELECT *  
FROM shippers;
```

Result

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

Notes and Observations

- A shipper_id is assigned automatically for the new record without specifying it.

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

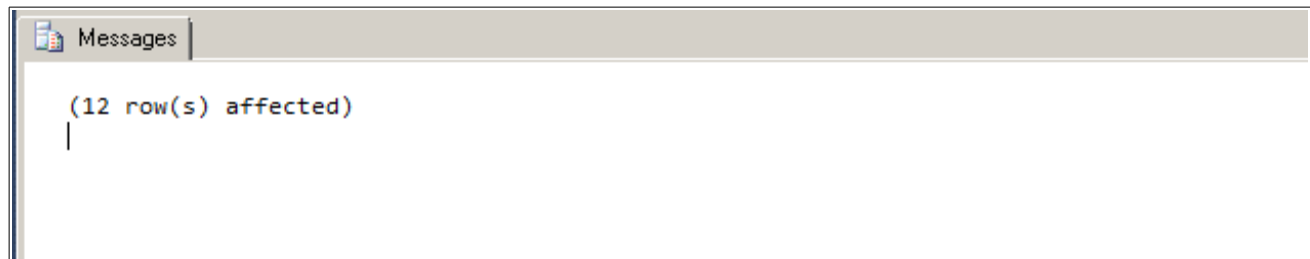
3.5 Part C Question 5

Using the UPDATE statement, increase 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.

Query

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

Results



Notes and Observations

- Had the WHERE clause been forgotten, all records in the table would be updated.

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

3.6 Part C 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.

Query

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

Result



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

3.7 Part C 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.

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 Paillard	420.0000
10001	2001-05-13 00:00:00.000	40	Mère Paillard	736.0000
10001	2001-05-13 00:00:00.000	59	Mère Paillard	440.0000
10001	2001-05-13 00:00:00.000	64	Mère Paillard	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	Comidas preparadas	588.0000
(540 row(s) affected)				

Query

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

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

```
        name,  
        order_cost  
FROM vw_order_cost  
WHERE order_id BETWEEN 10000 AND 10200;  
GO
```

Result (rows 1 to 12 from 540 rows)

	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 få HB	437.50
7	10002	2001-05-14 00:00:00.000	39	Folk och få HB	324.00
8	10002	2001-05-14 00:00:00.000	71	Folk och få 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	Vaffeljernet	4332.65
11	10004	2001-05-16 00:00:00.000	63	Vaffeljernet	263.40
12	10005	2001-05-20 00:00:00.000	62	Wartian Herkku	295.80

Results	Messages
(540 row(s) affected)	

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

3.8 Part C 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.

employee_id	Name	birth_date
5	Buchanan, Steven	1955.03.04
7	King, Robert	1960.05.29
9	Dodsworth, Anne	1966.01.27
(3 row(s) affected)		

Query

```
CREATE VIEW vw_list_employees (employee_id, name, birth_date)
AS
SELECT employee.employee_id,
        employee.last_name+', '+employee.first_name,
        CONVERT(CHAR(10),employee.birth_date,102)
FROM employee
GO

SELECT employee_id,
        name,
        birth_date
FROM vw_list_employees
WHERE employee_id IN (5,7,9);
GO
```


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

Results

Results		Messages	
	employee_id	name	birth_date
1	5	Buchanan, Steven	1955.03.04
2	7	King, Robert	1960.05.29
3	9	Dodsworth, Anne	1966.01.27

Results		Messages	
(3 row(s) affected)			

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

3.9 Part C 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.

order_id	customer_id	customer_name	city	country	shipped_date
10308	ANATR	Ana Trujillo Emparedados y helados	México D.F.	Mexico	Aug 18 2002
10365	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Oct 26 2002
10137	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Jan 22 2002
10142	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	Jan 8 2002
10218	ANTON	Antonio Moreno Taquería	México D.F.	Mexico	May 25 2002
...					
10344	WHITC	White Clover Markets	Seattle	United States	Sep 29 2002
10269	WHITC	White Clover Markets	Seattle	United States	Jul 3 2002
10374	WOLZA	Wolski Zajazd	Warszawa	Poland	Nov 2 2002
(293 row(s) affected)					

Query

```
CREATE VIEW vw_all_orders (order_id, customer_id, customer_name, city, country, shipped_date)
/* CREATE VIEW cannot be combined with other statements in batch */
AS
SELECT orders.order_id,
       customers.customer_id,
       customers.name,
       customers.city,
       customers.country,
       orders.shipped_date
FROM orders
```

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

```
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 'January 1, 2002' AND 'December 31, 2002'
ORDER BY customer_name, country;
GO
```

Result (rows 1 to 12 from 293 rows)

	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	10384	BERGS	Berglunds snabbköp	Luleå	Sweden	Nov 13 2002
10	10278	BERGS	Berglunds snabbköp	Luleå	Sweden	Jul 10 2002
11	10280	BERGS	Berglunds snabbköp	Luleå	Sweden	Aug 6 2002
12	10158	BERGS	Berglunds snabbköp	Luleå	Sweden	Feb 4 2002

(293 row(s) affected)

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

3.10 Part C 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.

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	Macauley Products Company	8	Northwoods Cranberry Sauce
15	Campbell Company	55	Pâté chinois
(77 row(s) affected)			

Query

```
CREATE VIEW vw_product_by_supplier (supplier_id,supplier_name,product_id,product_name)
AS
SELECT suppliers.supplier_id,
        suppliers.name,
        products.product_id,
        products.name
FROM products
INNER JOIN suppliers ON products.supplier_id=suppliers.supplier_id
GO

SELECT supplier_id,
        supplier_name,
        product_id,
        product_name
```

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

```
FROM vw_product_by_supplier;  
GO
```

Notes and Observations

- Assigning the VIEW its own field names is redundant as it inherits the names anyway if not provided.

Result (rows 1 to 12 of 77)

	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	Armstrong Company	11	Queso Cabrales
12	5	Armstrong Company	12	Queso Manchego La Pastora

Results	Messages
(77 row(s) affected)	

4 Part D - Stored Procedures and Triggers

4.1 Part D 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.

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)				

Query

```
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
```

4 Part D - Stored Procedures and Triggers

Result

	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

Results	Messages
(6 row(s) affected)	

4 Part D - Stored Procedures and Triggers

4.2 Part D 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.

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)

Query

```
CREATE PROCEDURE sp_orders_by_dates
(
    @start VARCHAR(30),
    @end VARCHAR(30)
)
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 orders.shipped_date BETWEEN @start AND @end;
```


4 Part D - Stored Procedures and Triggers

GO

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

GO

Notes and Observations

- SQL appears to be flexible about inputs of the type VARCHAR for searches with regards to date.

Results (rows 1 to 14 of 188 rows)

	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

Results	Messages
(188 row(s) affected)	

4 Part D - Stored Procedures and Triggers

4.3 Part D 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.

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)			

Query

```
CREATE PROCEDURE sp_product_listing
(
    @product VARCHAR(30),
    @month VARCHAR(10),
    @year VARCHAR(10)
)
AS
    SELECT 'product_name'=products.name,
           products.unit_price,
           products.quantity_in_stock,
           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
    DATEPART(year, orders.order_date)=@year;
GO
EXECUTE sp_product_listing '%Jack%', 'June', 2001;
GO
```

4 Part D - Stored Procedures and Triggers

Results

	product_name	unit_price	quantity_in_stock	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

Results	Messages
<pre>(4 row(s) affected) </pre>	

4 Part D - Stored Procedures and Triggers

4.4 Part D Question 4

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

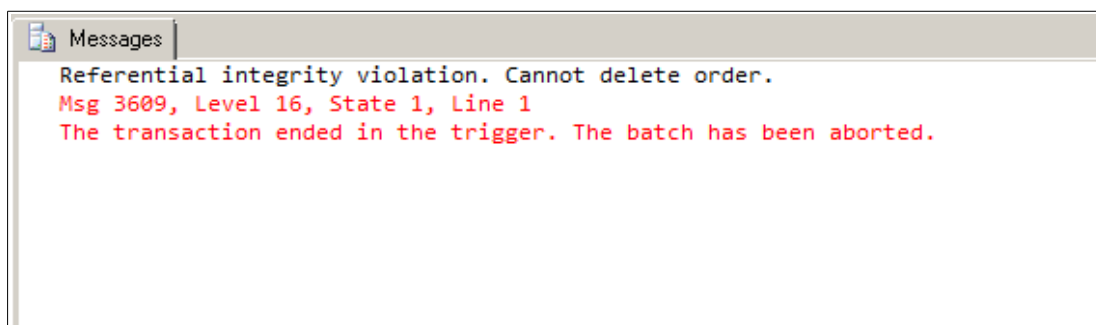
```
DELETE orders
WHERE order_id = 10000
```

Query

```
CREATE TRIGGER tr_delete_orders
ON orders
INSTEAD OF DELETE
AS
DECLARE @testid VARCHAR(15)
SELECT @testid=order_id
FROM DELETED;
IF EXISTS
(
SELECT order_details.order_id
FROM order_details
WHERE order_details.order_id=@testid
)
BEGIN
    PRINT 'Referential integrity violation. Cannot delete order.'
    ROLLBACK TRANSACTION
END;
GO

DELETE orders
WHERE order_id=10000;
GO
```

Results



4 Part D - Stored Procedures and Triggers

4.5 Part D Question 5

Create an INSERT and UPDATE trigger called tr_check_qty on the order_details table to only allow orders of products that have a quantity in stock greater than or equal to the units ordered. Run the following query to verify your trigger.

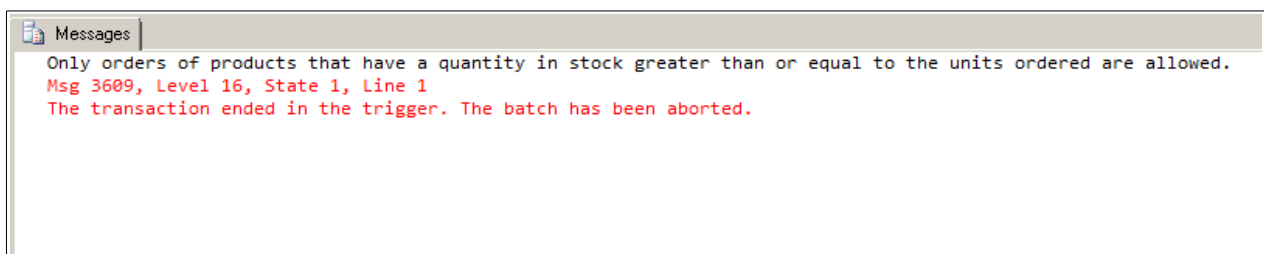
```
UPDATE order_details
SET quantity = 30
WHERE order_id = '10044'
      AND product_id = 7
```

Query

```
CREATE TRIGGER tr_check_qty
ON order_details
FOR INSERT, UPDATE
AS
IF EXISTS
(
SELECT
    order_id
FROM INSERTED
INNER JOIN products ON INSERTED.product_id=products.product_id
WHERE INSERTED.quantity < products.quantity_in_stock
)
BEGIN
    PRINT 'Only orders of products that have a quantity in stock greater than or equal to
the units ordered are allowed.'
    ROLLBACK TRANSACTION
END;
GO

UPDATE order_details
SET quantity=30
WHERE order_id='10044';
GO
```

Result



4 Part D - Stored Procedures and Triggers

4.6 Part D Question 6

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

Query

```
CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE FROM customers
WHERE active=0;
GO

EXECUTE sp_del_inactive_cust;
GO
```

Result



Notes and Observations

- Back in 2.2, customers with no orders were given "0" as the "active" value

4 Part D - Stored Procedures and Triggers

4.7 Part D Question 7

Create a stored procedure called `sp_employee_information` to display the employee information for a particular employee. The employee id will be an input parameter for the stored procedure. Run the stored procedure displaying information for employee id of 5. The stored procedure should produce the result set listed below.

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

(1 row(s) affected)

Query

```
CREATE PROCEDURE sp_employee_information
(
    @emp_id_input VARCHAR(15)
)
AS
    SELECT employee_id,
           last_name,
           first_name,
           address,
           city,
           province,
           postal_code,
           phone,
           birth_date
    FROM employee
    WHERE employee_id=@emp_id_input
GO

EXECUTE sp_employee_information 5;
GO
```

Result

	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	1955-03-04 00:00:00.000

4 Part D - Stored Procedures and Triggers

4.8 Part D Question 8

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

product_id	name	address	city	province	qty	reorder_level
2	Edward's Products Ltd.	1125 Howe Street	Vancouver	BC	17	25
3	Edward's Products Ltd.	1125 Howe Street	Vancouver	BC	13	25
5	New Orlean's Spices Ltd.	1040 Georgia Street	West Vancouver	BC	0	0
11	Armstrong Company	1638 Derwent Way	Richmond	BC	22	30
17	Steveston Export Company	2951 Moncton Street	Richmond	BC	0	0
...						
68	Dare Manufacturer Ltd.	1603 3rd Avenue	West Burnaby	BC	6	15
70	Steveston Export Company	2951 Moncton Street	Richmond	BC	15	30
74	Yves Delorme Ltd.	3050 Granville Street	New Westminster	BC	4	5
(23 row(s) affected)						

4 Part D - Stored Procedures and Triggers

Query

```
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 (rows 1 to 20 of 23)

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

4 Part D - Stored Procedures and Triggers

4.9 Part D Question 9

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

product_id	name	alternate_name	unit_price
13	Konbu	Kelp Seaweed	6.30
19	Teatime Chocolate Biscuits	Teatime Chocolate Biscuits	9.66
23	Tunnbrød	Thin Bread	9.45
45	Røgede sild	Smoked Herring	9.975
47	Zaanse koeken	Zaanse Cookies	9.975
52	Filo Mix	Mix for Greek Filo Dough	7.35
54	Tourtière	Pork Pie	7.8225
75	Rhénish Klosterbier	Rhénish Beer	8.1375
(8 row(s) affected)			



Query

```
CREATE PROCEDURE sp_unit_prices
(
    @unit_price_low_end VARCHAR(15),
    @unit_price_high_end VARCHAR(15)
)
AS
    SELECT product_id,
           name,
           alternate_name,
           unit_price
    FROM products
    WHERE unit_price BETWEEN @unit_price_low_end AND @unit_price_high_end
GO

EXECUTE sp_unit_prices 5, 10;
```

4 Part D - Stored Procedures and Triggers

Results

 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

5 Query file

```

/***** Drop previous database *****/

USE master;                                --select master which has all the
databases
GO

IF EXISTS                                  --checking for the existence of
'Cus_Orders' database and deletes it if found
(
    SELECT *
    FROM sysdatabases
    WHERE name='Cus_Orders'
)
BEGIN
    PRINT('Dropping database...')
    DROP DATABASE Cus_Orders;
END
GO

/***** PART A *****/

USE master;
GO

/*#1*/

CREATE DATABASE Cus_Orders;
GO

USE Cus_Orders;
GO

/*#2*/

CREATE TYPE uniqid FROM CHAR(5) NOT NULL;
CREATE TYPE intid FROM int NOT NULL;
GO

/*#3*/
/* TABLE CREATION */

CREATE TABLE customers
(
    customer_id uniqid,
    name VARCHAR(50),
    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),

```

5 Query file

```
        fax VARCHAR(20)
    );

CREATE TABLE orders
(
    order_id intid,
    customer_id uniqid,
    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),
    alternate_name VARCHAR(40),
    quantity_per_unit VARCHAR(25),
    unit_price money,
    quantity_in_stock int,
    units_on_stock int,
    reorder_level int
);

CREATE TABLE shippers
(
    shipper_id int IDENTITY(1,1) NOT NULL,
    name VARCHAR(20) NOT NULL
);

CREATE TABLE suppliers
(
    supplier_id int IDENTITY(1,1) NOT NULL,
    name VARCHAR(40) NOT NULL,
    address VARCHAR(30),
    city VARCHAR(20),
    province CHAR(2)
```

5 Query file

```
);

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

/*#4*/
/*CREATE PK's*/

ALTER TABLE customers
ADD PRIMARY KEY (customer_id);
GO

ALTER TABLE orders
ADD PRIMARY KEY (order_id);
GO

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

ALTER TABLE products
ADD PRIMARY KEY (product_id);
GO

ALTER TABLE shippers
ADD PRIMARY KEY (shipper_id);
GO

ALTER TABLE suppliers
ADD PRIMARY KEY (supplier_id);
GO

ALTER TABLE titles
ADD PRIMARY KEY (title_id);
GO

/* ADD FK's */

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

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

ALTER TABLE order_details
ADD CONSTRAINT FK_orders_order_details FOREIGN KEY(order_id)
```

5 Query file

```
REFERENCES orders(order_id);
GO

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

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

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

/*#5*/
/* Other constraints */

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

ALTER TABLE orders
ADD CONSTRAINT default_date
    DEFAULT(DATEADD(DAY,10,GETDATE())) FOR order_date;
GO

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

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

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

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

/*#6*/
/* Load data into tables */
```

5 Query file

```
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,
```


5 Query file

```
        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 *****/

/*#1*/
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;

/*#2*/
ALTER TABLE customers
ADD active bit;
GO

ALTER TABLE customers
ADD CONSTRAINT default_active
    DEFAULT (1) FOR active;
GO

UPDATE customers
SET active=1
WHERE EXISTS
(
    SELECT *
    FROM orders
    WHERE orders.customer_id=customers.customer_id
)
GO

UPDATE customers
SET active=0
WHERE NOT EXISTS
(
    SELECT *
    FROM orders
    WHERE orders.customer_id=customers.customer_id
)
GO
```

5 Query file

/*#3*/

```
SELECT 'order_id'=orders.order_id,
       'product_name'=products.name,
       'customer_name'=customers.name,
       'order_date'= CONVERT(CHAR(11),orders.order_date,0),
       'new_shipped_date'= CONVERT(CHAR(11),DATEADD(DAY,7,orders.order_date)),
       'order_cost'= order_details.quantity*products.unit_price
FROM order_details
INNER JOIN orders 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
WHERE orders.order_date BETWEEN 'January 1, 2001' AND 'December 31, 2001';
```

/*#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;
```

/*#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;
```

/*#6*/

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

/*#7*/

```
SELECT orders.order_id,
       customers.name,
       customers.contact_name,
       'shipped_date'=CONVERT(CHAR(11),orders.shipped_date, 100),
       'elapsed'=DATEDIFF(YEAR, orders.shipped_date, 'January 1, 2008')
```

5 Query file

```
FROM orders
INNER JOIN customers ON orders.customer_id=customers.customer_id
WHERE orders.shipped_date IS NOT NULL
ORDER BY orders.order_id, 'elapsed';

/*#8*/

SELECT 'name' = SUBSTRING(name,1,1),
       'total' = COUNT(name)
FROM customers
WHERE SUBSTRING(name,1,1) LIKE '[^S]'
GROUP BY SUBSTRING(name,1,1)
HAVING COUNT(name) >= 2;

/*#9*/

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

/*#10*/

SELECT products.product_id,
       products.name,
       products.quantity_per_unit,
       products.unit_price
FROM products
WHERE products.name LIKE '%tofu%'
UNION ALL
SELECT products.product_id,
       products.name,
       products.quantity_per_unit,
       products.unit_price
FROM products
WHERE products.name LIKE '%chef%'
ORDER BY products.name;

/***** PART C *****/

/*#1*/

CREATE TABLE employee
(
    employee_id int NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    first_name VARCHAR(15) NOT NULL,
    address VARCHAR(30),
    city VARCHAR(20),
```

5 Query file

```
        province CHAR(2),
        postal_code VARCHAR(7),
        phone VARCHAR(10),
        birth_date datetime
    );
GO

/*#2*/

ALTER TABLE employee
ADD PRIMARY KEY (employee_id);
GO

/*#3*/

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

ALTER TABLE orders
ADD CONSTRAINT FK_employee_orders FOREIGN KEY(employee_id)
REFERENCES employee(employee_id);
GO

/*#4*/

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

SELECT *
FROM shippers;

/*#5*/

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

/*#6*/

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

/*#7*/
```

5 Query file

```
CREATE VIEW vw_order_cost(order_id, order_date, product_id, name, order_cost)
AS
SELECT orders.order_id,
       orders.order_date,
       products.product_id,
       customers.name,
       'order_cost'=order_details.quantity*products.unit_price
FROM order_details
INNER JOIN orders 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 order_id,
       order_date,
       product_id,
       name,
       order_cost
FROM vw_order_cost
WHERE order_id BETWEEN 10000 AND 10200;
GO

/**8*/

CREATE VIEW vw_list_employees (employee_id, name, birth_date)
AS
SELECT employee.employee_id,
       employee.last_name+', '+employee.first_name,
       CONVERT(CHAR(10),employee.birth_date,102)
FROM employee
GO

SELECT employee_id,
       name,
       birth_date
FROM vw_list_employees
WHERE employee_id IN (5,7,9);
GO

/**9*/

CREATE VIEW vw_all_orders (order_id, customer_id, customer_name, city, country, shipped_date)
/* CREATE VIEW cannot be combined with other statements in batch */
AS
SELECT orders.order_id,
       customers.customer_id,
       customers.name,
       customers.city,
       customers.country,
       orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id=customers.customer_id;
```

5 Query file

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
```

/*#10*/

```
CREATE VIEW vw_product_by_supplier (supplier_id, supplier_name, product_id, product_name)
AS
SELECT suppliers.supplier_id,
       suppliers.name,
       products.product_id,
       products.name
FROM products
INNER JOIN suppliers ON products.supplier_id=suppliers.supplier_id
GO
```

```
SELECT supplier_id,
       supplier_name,
       product_id,
       product_name
FROM vw_product_by_supplier;
GO
```

/******Part D******/

/*#1*/

```
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
```

5 Query file

```
/*#2*/

CREATE PROCEDURE sp_orders_by_dates
(
    @start VARCHAR(30),
    @end VARCHAR(30)
)
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 orders.shipped_date BETWEEN @start AND @end;

GO

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

/*#3*/

CREATE PROCEDURE sp_product_listing
(
    @product VARCHAR(30),
    @month VARCHAR(10),
    @year VARCHAR(10)
)
AS
    SELECT 'product_name'=products.name,
           products.unit_price,
           products.quantity_in_stock,
           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
    DATEPART(year, orders.order_date)=@year;

GO

EXECUTE sp_product_listing '%Jack%', 'June', 2001;
GO

/*#4*/

CREATE TRIGGER tr_delete_orders
ON orders
INSTEAD OF DELETE
AS
DECLARE @testid VARCHAR(15)
SELECT @testid=order_id
FROM DELETED;
```

5 Query file

```
IF EXISTS
(
SELECT order_details.order_id
FROM order_details
WHERE order_details.order_id=@testid
)
BEGIN
    PRINT 'Referential integrity violation. Cannot delete order.'
    ROLLBACK TRANSACTION
END;
GO

DELETE orders
WHERE order_id=10000;
GO

/*#5*/

CREATE TRIGGER tr_check_qty
ON order_details
FOR INSERT, UPDATE
AS
IF EXISTS
(
SELECT
    order_id
FROM INSERTED
INNER JOIN products ON INSERTED.product_id=products.product_id
WHERE INSERTED.quantity < products.quantity_in_stock
)
BEGIN
    PRINT 'Only orders of products that have a quantity in stock greater than or equal to
the units ordered are allowed.'
    ROLLBACK TRANSACTION
END;
GO

UPDATE order_details
SET quantity=30
WHERE order_id='10044';
GO

/*#6*/

CREATE PROCEDURE sp_del_inactive_cust
AS
DELETE FROM customers
WHERE active=0;
GO

EXECUTE sp_del_inactive_cust;
GO

/*#7*/
```


5 Query file

```
CREATE PROCEDURE sp_employee_information
(
    @emp_id_input VARCHAR(15)
)
AS
    SELECT employee_id,
           last_name,
           first_name,
           address,
           city,
           province,
           postal_code,
           phone,
           birth_date
    FROM employee
    WHERE employee_id=@emp_id_input
GO

EXECUTE sp_employee_information 5;
GO

/*#8*/

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

EXECUTE sp_reorder_qty 5;
GO

/*#9*/

CREATE PROCEDURE sp_unit_prices
(
    @unit_price_low_end VARCHAR(15),
    @unit_price_high_end VARCHAR(15)
)
AS
    SELECT product_id,
           name,
           alternate_name,
```

5 Query file

```
        unit_price
FROM products
WHERE unit_price BETWEEN @unit_price_low_end AND @unit_price_high_end
GO

EXECUTE sp_unit_prices 5, 10;
```