COMP1630 Practical PROJECT

Student: Nero Deng

This project was created in order to teach students, how to use SQL Server. By providing questions separated in four major sections, students then must approach the questions by using Microsoft SQL Server.

Table Of Content

Part *A* - Practical Section --Database and Tables

Before we start we should removed previous cus_orders database

```
--- Removed Pervious Cus_Order DataBase
USE master
GO

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

--Now we're ready to go.

Question 1. Create a database called cus_orders.

```
/* Part A */
/*1.*/  CREATE DATABASE Cus_Orders;
go
USE Cus_Orders;
```

Result :

```
Messages
Command(s) completed successfully.
```

Question 2. Create user defined datatypes for similar primary key attribute columns, to ensure the same data type, length and nullability.

```
/*2.*/
CREATE TYPE csid FROM char(5) NOT NULL;
CREATE TYPE orid FROM int NOT NULL;
DROP TYPE csid;
DROP TYPE orid;
CREATE TYPE csid FROM char(5) NOT NULL;
CREATE TYPE orid FROM int NOT NULL;
```

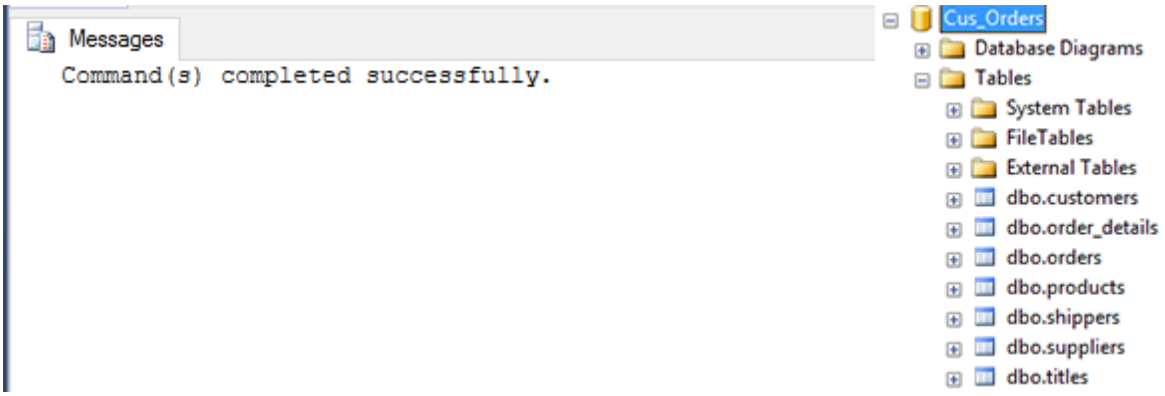
Result:

```
Messages
Command(s) completed successfully.
```

Question 3. Create customers table, orders table, order_details table, products table, shippers table, and titles table.

```
/*3.*/
CREATE TABLE customers(
customer_id     csid,
name varchar(50) NOT NUll,
contact_name varchar(30),
title_id char(3),
address varchar(50),
city varchar(20),
region varchar(15),
country_code varchar(10),
country varchar(15),
phone varchar(20),
fax varchar(20)
);
go
CREATE TABLE orders(
order_id orid,
customer_id csid,
employee_id int NOT Null,
shipping_name varchar(50),
```

Result:



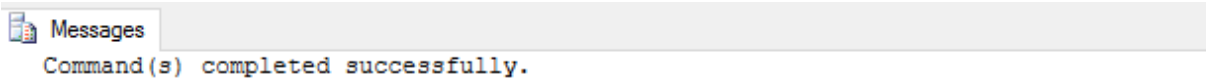
Question 4. Set the primary keys and foreign keys for the tables.

```
/*4.*/
ALTER TABLE customers
ADD PRIMARY KEY(customer_id);

ALTER TABLE titles
ADD PRIMARY KEY (title_id);

ALTER TABLE customers
ADD CONSTRAINT fk_customers_titles
FOREIGN KEY (title_id)
```

Result:



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

```
/*5.*/
ALTER TABLE Customers
ADD CONSTRAINT default_country
    DEFAULT ('Canada') FOR country;
go
ALTER TABLE orders
ADD CONSTRAINT default_date
DEFAULT (DATEADD (DAY,10,'required_date')) FOR required_date;
go
ALTER TABLE order_details
ADD CONSTRAINT quantity CHECK (quantity >=1);

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

Result:

```
DEFAULT ('BC') FOR province;
```

110 %

Messages

```
Command(s) 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)

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

```
/* #6 */
BULK INSERT titles
FROM 'C:\TextFiles\titles.txt'
WITH (
                CODEPAGE=1252,
        DATAFILETYPE = 'char',
        FIELDTERMINATOR = '\t',
        KEEPNULLS,
        ROWTERMINATOR = '\n'
      )
go
```

Result:

```
--- Removed Pervious Cus_Order DataBase
USE master
GO

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

Part *B* - Practical Section -- 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.

```
/* Part B */
SELECT customer_id, name, city, country
FROM customers
ORDER BY customer_id;
go
```

Result:

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

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (55) | 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.

```
ALTER TABLE customers
ADD active smallint
CONSTRAINT Ch_customers_active
    CHECK (active IN ('1', '0'))
DEFAULT '1';
```

Result:

110 %

Messages

```
Command(s) completed successfully.
```

110 %

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (54) | Cus_Orders | 00:00:00 | 0 rows

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.

```
SELECT orders.order_id,
        'product_name' = products.name,
        'customer_name' = customers.name,
        CONVERT(VARCHAR(10), order_date, 100) AS order_date,
        'new_shipped_date'=DATEADD (DAY, 7, (orders.shipped_date)),
        order_details.quantity*products.unit_price AS 'order_cost'
FROM orders
INNER JOIN order_details ON orders.order_id = order_details.order_id
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE order_date BETWEEN 'January 1, 2001' AND 'December 31, 2001'
ORDER BY order_id;
```

Result:

110 %

Results | Messages

	order_id	product_name	customer_name	order_date	new_shipped_date	order_cost
1	10000	Alice Mutton	Franchi S.p.A.	May 10 200	2001-05-22 00:00:00.000	156.00
2	10001	NuNuCa Nuß-Nougat-Creme	Mère Paillarde	May 13 200	2001-05-30 00:00:00.000	420.00
3	10001	Boston Crab Meat	Mère Paillarde	May 13 200	2001-05-30 00:00:00.000	736.00
4	10001	Raclette Courdavault	Mère Paillarde	May 13 200	2001-05-30 00:00:00.000	440.00
5	10001	Wimmers gute Semmelknö...	Mère Paillarde	May 13 200	2001-05-30 00:00:00.000	498.75

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (54) | 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. 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.

```
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 shipped_date IS NULL;
```

Result:

Results | Messages

	customer_id	name	phone	order_id	order_date
15	LILAS	LILA-Supermercado	(9) 331-6954	11071	2004-03-29 00:00:00.000
16	ERNSH	Ernst Handel	7675-3425	11072	2004-03-29 00:00:00.000
17	PERIC	Pericles Comidas clásicas	(5) 552-3745	11073	2004-03-29 00:00:00.000
18	SIMOB	Simons bistro	31 12 34 56	11074	2004-03-30 00:00:00.000
19	RICSU	Richter Supermarkt	0897-034214	11075	2004-03-30 00:00:00.000
20	BONAP	Bon app'	91.24.45.40	11076	2004-03-30 00:00:00.000

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (53) | 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. The query should produce the result set listed below.

```
SELECT customers.customer_id,
        customers.name,
        customers.city,
        titles.description
FROM customers
INNER JOIN titles ON customers.title_id = titles.title_id
WHERE region IS NULL
ORDER BY customer_id;
```

Result:

110 %

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. | (local) (13.0 RC2) | Viper-PC\Viper (54) | 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. The query should produce the result set listed below.

```sql
/* 6. */
SELECT suppliers.name AS 'supplier_name',
        products.name AS 'product_name',
        products.reorder_level,
        products.quantity_in_stock
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE reorder_level > quantity_in_stock
ORDER BY suppliers.name;
```

Result:

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. | (local) (13.0 RC2) | Viper-PC\Viper (55) | 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. The query should produce the result set listed below.

```sql
/*B 7. */
SELECT orders.order_id,
        customers.name,
        customers.contact_name,
        CONVERT(varchar(10), orders.shipped_date, 100) AS 'shipped_date',
        'elapsed' = DATEDIFF(Year,orders.shipped_date, 'Jan 1 2008')
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
WHERE orders.shipped_date IS NOT NULL;
go
```

Result:

Results | Messages

	order_id	name	contact_name	shipped_date	elapsed
1	10000	Franchi S.p.A.	Paolo Accorti	May 15 200	7
2	10001	Mère Paillarde	Jean Fresnière	May 23 200	7
3	10002	Folk och fä HB	Maria Larsson	May 17 200	7
4	10003	Simons bistro	Jytte Petersen	May 24 200	7
5	10004	Vaffeljernet	Palle Ibsen	May 20 200	7

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (55) | 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.

```
/*B 8. */
SELECT name = SUBSTRING(name,1,1), 'total' = COUNT(name)
FROM customers
GROUP BY SUBSTRING(name,1,1)
HAVING COUNT(name) >= 2 AND SUBSTRING(name,1,1) != 'S';
go
```

Result :

Results | Messages

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

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (55) | 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 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.

```
/*B 9. */
SELECT order_details.order_id,
       order_details.quantity,
       products.product_id,
       products.reorder_level,
       suppliers.supplier_id
FROM order_details
INNER JOIN products ON order_details.product_id = products.product_id
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE order_details.quantity > 100
ORDER BY order_details.order_id;
go
```

Result:

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. | (local) (13.0 RC2) | Viper-PC\Viper (55) | 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.

```
/*B 10. */
SELECT product_id,
        name,
        quantity_per_unit,
        unit_price
FROM products
WHERE name  LIKE '%tofu%' OR name LIKE '%chef%'
ORDER BY product_id;
```

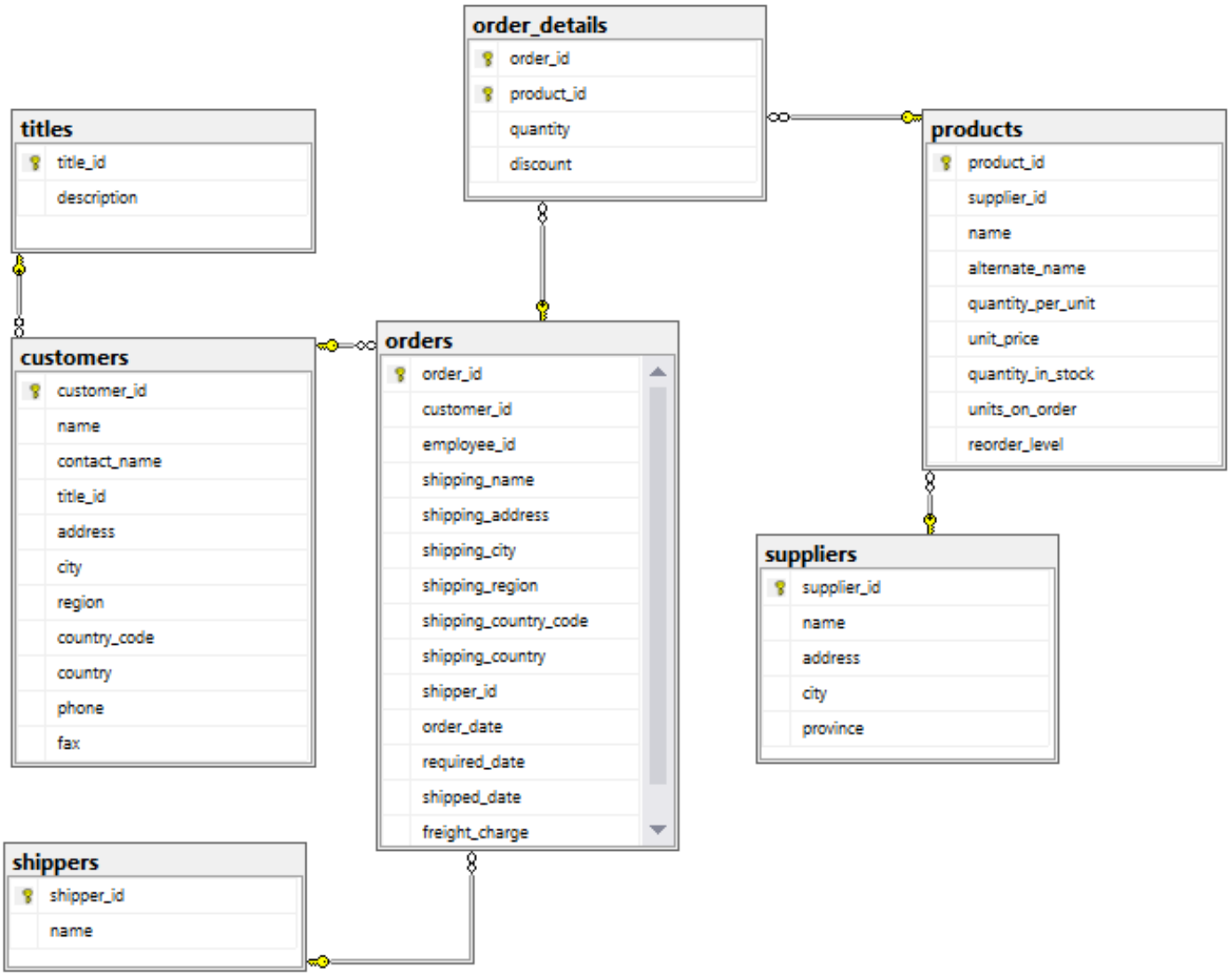
Result:

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	14	Tofu	40 - 100 g pkgs.	23.25
4	74	Longlife Tofu	5 kg pkg.	10.00

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (55) | Cus_Orders | 00:00:00 | 4 rows

Part B- Diagram Review



Part *C* - Practical Section --Insert, Update, Delete and Views

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
/* Part C. */
/*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),
 province char(2),
 postal_code varchar(7),
 phone varchar(10),
 birth_date datetime NOT NULL);
go
```

Result:

```
110 %
Messages
Command(s) completed successfully.
110 %
Query executed successfully.   (local) (13.0 RC2)   Viper-PC\Viper (51)   Cus_Orders   00:00:00   0 rows
```

Question 2. The primary key for the employee table should be the employee id.

```
/*C 2. */
ALTER TABLE employee
ADD PRIMARY KEY (employee_id);
go
```

Result:

```
Messages
Command(s) completed successfully.
```

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.
<u>(This question is completed by two separate events)</u>

Question 3. A

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

Result:

```
Messages
(9 row(s) affected)
110 %
Query executed successfully.   (local) (13.0 RC2)   Viper-PC\Viper (51)   Cus_Orders   00:00:00   0 rows
```

Question 3. B

```
ALTER TABLE orders
ADD CONSTRAINT fk_employee_orders
FOREIGN KEY (employee_id)
REFERENCES employee(employee_id);
go
```

Result:

```
Messages
Command(s) completed successfully.
```

executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (51) | Cus_Orders | 00:00:00 | 0 rows

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

```
/*C 4. */
INSERT INTO shippers(name)
VALUES ('Quick Express')
```

Result:

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

110 %

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (52) | Cus_Orders | 00:00:00 | 0 rows

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
/*C 5. */
UPDATE products
SET unit_price = unit_price*1.05
WHERE unit_price BETWEEN 5 AND 10;
```

Result:

```
Messages

(12 row(s) affected)

110 %
Query executed successfully.   (local) (13.0 RC2)   Viper-PC\Viper (52)   Cus_Orders   00:00:00   0 rows
```

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
/*C 6.*/
UPDATE customers
SET fax='unknown'
WHERE fax is null;
```

Result:

```
110 %
Messages

(22 row(s) affected)

110 %
Query executed successfully.   (local) (13.0 RC2)   Viper-PC\Viper (52)   Cus_Orders   00:00:00   0 rows
```

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 tble, and the order cost. To calculate the cost of the orders, use the formula (order_details.quantity * products.unit_price). Run the view for the order ids between 10000 and 10200. The view should produce the result set listed below.

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

Result:

Results | Messages

	order_id	order_date	product_id	name	order_cost
1	10021	2001-06-14 00:00:00.000	1	Ernst Handel	1080.00
2	10043	2001-07-22 00:00:00.000	1	LINO-Delicateses	720.00
3	10065	2001-08-28 00:00:00.000	1	Save-a-lot Markets	990.00
4	10071	2001-09-06 00:00:00.000	1	Hungry Owl All-Night Grocers	450.00
5	10077	2001-09-17 00:00:00.000	1	La maison d'Asie	504.00
6	10101	2001-10-28 00:00:00.000	1	Antonio Moreno Taquería	144.00
7	10156	2002-01-28 00:00:00.000	1	Richter Supermarkt	450.00

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (52) | 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.

```sql
/*C 8.*/
CREATE VIEW vw_list_employees
AS SELECT employee.employee_id,
            employee.first_name + employee.last_name AS 'NAME',
            CONVERT (varchar(11),employee.birth_date,102) AS 'birth_day'
FROM employee
WHERE employee_id = 5 OR employee_id = 7 OR employee_id = 9;
go
SELECT * FROM vw_list_employees;
```

Result:

Results | Messages

	employee_id	NAME	birth_day
1	5	StevenBuchanan	1955.03.04
2	7	RobertKing	1960.05.29
3	9	AnneDodsworth	1966.01.27

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (52) | 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 January1, 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.

```
/*C 9.*/
CREATE VIEW vw_all_orders
AS SELECT orders.order_id,
            customers.customer_id,
            'customer_name' = customers.name,
            customers.city,
            customers.country,
            orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;
go

SELECT order_id,
            customer_id,
            customer_name,
            city,
            country,
            CONVERT(varchar(11), shipped_date, 100) AS shipped_date
FROM vw_all_orders
WHERE shipped_date BETWEEN 'Jan 1,2002' AND 'Dec 31,2002'
go
```

Result:

110 %

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. | (local) (13.0 RC2) | Viper-PC\Viper (56) | 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.

```
/*C 10.*/
CREATE VIEW vw_supply_product
AS SELECT suppliers.supplier_id,
            'supplier_name' = suppliers.name,
            products.product_id,
            'product_name' = products.name
FROM suppliers
INNER JOIN products ON suppliers.supplier_id = products.supplier_id;
go

SELECT * FROM vw_supply_product;
go
```

Result:

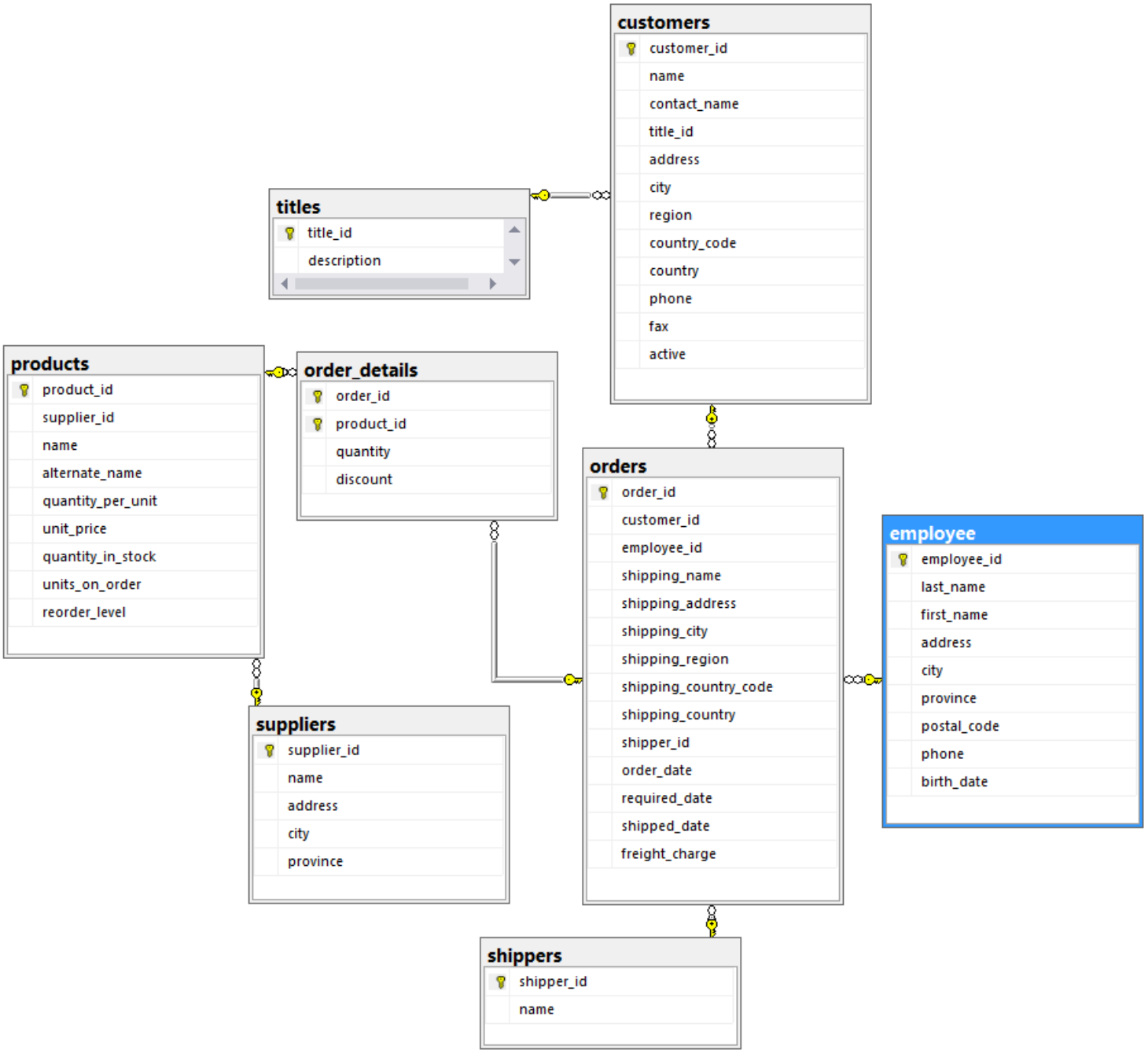
110 %

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. | (local) (13.0 RC2) | Viper-PC\Viper (56) | Cus_Orders | 00:00:00 | 77 rows

Part *C* - Updated Database Diagram



Part *D* - Practical Section --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.

```
/*D 1. */
CREATE PROCEDURE sp_customer_city
(
@city varchar(20)
)
AS SELECT customers.customer_id,
            customers.name,
            customers.address,
            customers.city,
            customers.phone
FROM customers
WHERE city=@city
go
EXECUTE sp_customer_city 'London'
```

Result:

Results | Messages

	customer_id	name	address	city	phone
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. | (local) (13.0 RC2) | Viper-PC\Viper (52) | 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.

```
/*D 2.*/
CREATE PROCEDURE sp_orders_by_dates
(
 @start datetime,
 @end datetime
)
AS SELECT orders.order_id,
            orders.customer_id,
            customers.name AS 'customer_name',
            shippers.name AS 'shipper_name',
            orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id
WHERE shipped_date BETWEEN @start AND @end;
go
EXECUTE sp_orders_by_dates 'Jan 1, 2003' , 'Jun 30, 2003';
go
```

Result:

110 %

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. | (local) (13.0 RC2) | Viper-PC\Viper (52) | 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.

```
/*D 3.*/   ---- This Question was a CHALLENGE ----
CREATE PROCEDURE sp_product_listing
(
@product varchar(50),
@month varchar(11),
@year int
)
AS SELECT products.name AS 'product_name',
            products.unit_price,
            products.quantity_in_stock,
            suppliers.name AS 'supplier_name'
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
INNER JOIN order_details ON products.product_id = order_details.product_id
INNER JOIN orders ON order_details.order_id = orders.order_id
WHERE products.name LIKE '%' +@product +'%'
AND DATENAME(MONTH, orders.order_date) = @month
AND DATENAME(YEAR, orders.order_date) = @year --Lightbuld--If It also linking to o

EXECUTE sp_product_listing 'Jack', June, 2001
go

DROP PROCEDURE sp_product_listing
```

Result:

```
110 %
Results
product_name                                unit_price          quantity_in_stock supplier_n
------------------------------------------- ------------------- ----------------- ----------
Jack's New England Clam Chowder             10.1325             85                Silver Spr
Jack's New England Clam Chowder             10.1325             85                Silver Spr
Jack's New England Clam Chowder             10.1325             85                Silver Spr
Jack's New England Clam Chowder             10.1325             85                Silver Spr
110 %
Query executed successfully.  (local) (13.0 RC2)  Viper-PC\Viper (52)  Cus_Orders  00:00:00  4 rows
```

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.

```
/*D 4. */
CREATE TRIGGER tr_delete_orders
ON orders
INSTEAD OF DELETE
AS
DECLARE @ord_id char(21)
SELECT @ord_id = order_id
FROM DELETED
IF EXISTS (SELECT order_id FROM order_details WHERE order_id = @ord_id)
BEGIN
    PRINT 'Deletions not allowed on orders table'
    ROLLBACK TRANSACTION
END;
go
DELETE orders
WHERE order_id = 10000;
go
```

Result:

```
110 %
Results
Deletions not allowed on orders table
Msg 3609, Level 16, State 1, Line 551
The transaction ended in the trigger. The batch has been aborted.
110 %
Query completed with errors.   (local) (13.0 RC2)   Viper-PC\Viper (52)   Cus_Orders   00:00:00   0 rows
```

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.

```
/*D 5. */
CREATE TRIGGER tr_check_qty
ON order_details
FOR INSERT, UPDATE
AS
DECLARE @qty_id char(21)
SELECT @qty_id = product_id
FROM inserted
IF (SELECT quantity_in_stock FROM products WHERE product_id = @qty_id) >=
(SELECT units_on_order FROM products WHERE product_id =@qty_id)
BEGIN
    PRINT 'Update amount exceed the amount of product in stock'
    ROLLBACK TRANSACTION
END;
go
UPDATE order_details
SET quantity = 30
WHERE order_id = '10044'
    AND product_id = 7;
go
```

Result:

```
110 %
Results
Update amount exceed the amount of product in stock
Msg 3609, Level 16, State 1, Line 570
The transaction ended in the trigger. The batch has been aborted.
110 %
Query completed with errors.   (local) (13.0 RC2)   Viper-PC\Viper (57)   Cus_Orders   00:00:00   0 rows
```

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.

```
/*D 6.*/
CREATE PROCEDURE sp_delinactive_cust
AS DELETE
FROM customers
WHERE customers.customer_id NOT IN (
    SELECT orders.customer_id
    FROM orders
)
EXECUTE sp_delinactive_cust
```

Result:

```
110 %
Results

(1 row(s) affected)

110 %
Query executed successfully.   (local) (13.0 RC2)   Viper-PC\Viper (57)   Cus_Orders   00:00:00   0 rows
```

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.

```
/*D 7. */
CREATE PROCEDURE sp_employee_information
(
    @emp_id int
)
AS SELECT employee.employee_id,
          employee.last_name,
          employee.first_name,
          employee.city,
          employee.province,
          employee.postal_code,
          employee.phone,
          employee.birth_date
FROM employee
WHERE employee_id = @emp_id

EXECUTE sp_employee_information @emp_id = '5'
```

Result:

```
110 %
Results
employee_id last_name                    first_name      city                 province po
----------- ---------------------------- --------------- -------------------- -------- --
5           Buchanan                     Steven          New Westminster      BC       V1

(1 row(s) affected)
110 %
Query executed successfully.   (local) (13.0 RC2)   Viper-PC\Viper (57)   Cus_Orders   00:00:00   1 rows
```

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.

```sql
/*D 8.*/
CREATE PROCEDURE sp_reorder_qty
(
    @unit int
)
AS SELECT products.product_id,
            suppliers.name,
            suppliers.address,
            suppliers.city,
            suppliers.province,
            products.quantity_in_stock AS 'qty',
            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'
```

Result:

```
110 %
Results
66          New Orlean's Spices Ltd.          1040 Georgia Street West       Vancouv
68          Dare Manufacturer Ltd.            1603 3rd Avenue West           Burnaby
70          Steveston Export Company          2951 Moncton Street            Richmon
74          Yves Delorme Ltd.                 3050 Granville Street          New Wes

(23 row(s) affected)
110 %
Query executed successfully.   (local) (13.0 RC2)   Viper-PC\Viper (57)   Cus_Orders   00:00:00   23 rows
```

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

```
/*D 9. */ --- See Challenge -----
CREATE PROCEDURE sp_unity_price
(
    @price1 money,
    @price2 money
)
AS SELECT products.product_id,
            products.name,
            products.alternate_name,
            products.unit_price
FROM products
WHERE unit_price BETWEEN @price1 AND @price2;
go
EXECUTE sp_unity_price '5','10';
go
```

Result:

Results

product_id	name	alternate_name
13	Konbu	Kelp Seaweed
52	Filo Mix	Mix for Greek Filo Dough
54	Tourtière	Pork Pie
75	Rhönbräu Klosterbier	Rhönbräu Beer

110 %

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (57) | Cus_Orders | 00:00:00 | 4 rows

Challenge Section

Part D Question 2 - Solved

```
/*D 2.*/
CREATE PROCEDURE sp_orders_by_dates
(
 @start datetime,
 @end datetime
)
AS SELECT orders.order_id,
            orders.customer_id,
            customers.name AS 'customer_name',
            shippers.name AS 'shipper_name',
            orders.shipped_date
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN shippers ON orders.shipper_id = shippers.shipper_id
WHERE shipped_date BETWEEN '@start' AND '@end';
go

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

Results | Messages

```
Msg 241, Level 16, State 1, Procedure sp_orders_by_dates, Line 6 [Batch Start Line 511]
Conversion failed when converting date and/or time from character string.
```

Why was it a challenge?

As you could see that the code itself was correct. However when I tried executing the procedure, it gave me an error instead, which gave me quite a headache. Without understanding what "conversion failed when converting date and/or time from character string" meant. I decided to google it and found similar errors, solutions, and guides:

http://stackoverflow.com/questions/14119133/conversion-failed-when-converting-date-and-or-time-from-character-string-while-i

http://www.sql-server-helper.com/error-messages/msg-241.aspx

http://forums.asp.net/t/1836463.aspx?Conversion+failed+when+converting+date+and+or+time+from+character+string+

Part *D* Question 3 - Solved

```sql
/*D 3.*/
CREATE PROCEDURE sp_product_listing
(
@product varchar(50),
@month DATETIME,
@year DATETIME
)
AS SELECT products.name AS 'product_name',
            products.unit_price,
            products.quantity_in_stock,
            suppliers.name AS 'supplier_name'
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE products.name = '%' +@product +'%'

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

10 %

Results

```
Msg 8114, Level 16, State 5, Procedure sp_product_listing, Line 0 [Batch Start Line 529]
Error converting data type nvarchar to datetime.
```

Initially I couldn't figure out the error message, but after revisiting the question, it actually requires another data file (orders).

```sql
/*D 3.*/
CREATE PROCEDURE sp_product_listing
(
@product varchar(50),
@month DATETIME,
@year DATETIME
)
AS SELECT products.name AS 'product_name',
            products.unit_price,
            products.quantity_in_stock,
            suppliers.name AS 'supplier_name'
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
WHERE products.name = '%' +@product +'%'
AND DATENAME(MONTH, orders.order_date) = @month
AND DATENAME(YEAR, orders.order_date) = @year

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

%

Results

```
Msg 4104, Level 16, State 1, Procedure sp_product_listing, Line 14 [Batch Start Line 515]
The multi-part identifier "orders.order_date" could not be bound.
Msg 4104, Level 16, State 1, Procedure sp_product_listing, Line 15 [Batch Start Line 515]
The multi-part identifier "orders.order_date" could not be bound.
```

%

Query completed with errors. | (local) (13.0 RC2) | Viper-PC\Viper (52) | Cus_Orders | 00:00:00 | 0 rows

And another error message saying that it could not be bound, which is similar to a error related the inner join examples. Then I just used inner join, suppliers to products, products to order_details, order_details to orders and got the order_date. Which given us the correct answer.

```
/*D 3.*/   ---- This Question was a CHALLENGE ----
CREATE PROCEDURE sp_product_listing
(
@product varchar(50),
@month varchar(11),
@year int
)
AS SELECT products.name AS 'product_name',
            products.unit_price,
            products.quantity_in_stock,
            suppliers.name AS 'supplier_name'
FROM products
INNER JOIN suppliers ON products.supplier_id = suppliers.supplier_id
INNER JOIN order_details ON products.product_id = order_details.product_id
INNER JOIN orders ON order_details.order_id = orders.order_id
WHERE products.name LIKE '%' +@product +'%'
AND DATENAME(MONTH, orders.order_date) = @month
AND DATENAME(YEAR, orders.order_date) = @year --Lightbuld--If It also linking to o

EXECUTE sp_product_listing 'Jack', June, 2001
go

DROP PROCEDURE sp_product_listing
```

Part *D* Question 9 - Unsolved

```
/*D 9. */ --- See Challenge -----
CREATE PROCEDURE sp_unity_price
(
    @price1 money,
    @price2 money
)
AS SELECT products.product_id,
            products.name,
            products.alternate_name,
            products.unit_price
FROM products
WHERE unit_price BETWEEN @price1 AND @price2;
go
EXECUTE sp_unity_price '5','10';
go
```

110 %

Results

product_id	name	alternate_name
13	Konbu	Kelp Seaweed
52	Filo Mix	Mix for Greek Filo Dough
54	Tourtière	Pork Pie
75	Rhönbräu Klosterbier	Rhönbräu Beer

110 %

Query executed successfully. | (local) (13.0 RC2) | Viper-PC\Viper (57) | Cus_Orders | 00:00:00 | 4 rows

Creating the procedure

Step 1: Create the procedure
Step 2: Create the variables @price1 and @price2
Step 3: Select the listed column
Step 4: State a where clause with the two variables for executing the two prices

Executing the procedure

Step 1: Execute the procedure "name", price1 and price2

The steps above which i think it's correct but I only received 4 results.
The results 13, 52, 54, 75 which all within the range of 5 and 10 dollar, and the other 4 (19, 23, 45, 47) provided in the project description is also within the range of 5 and 10 dollar.

Difficulty Level 4 out of 5