Comp 1630

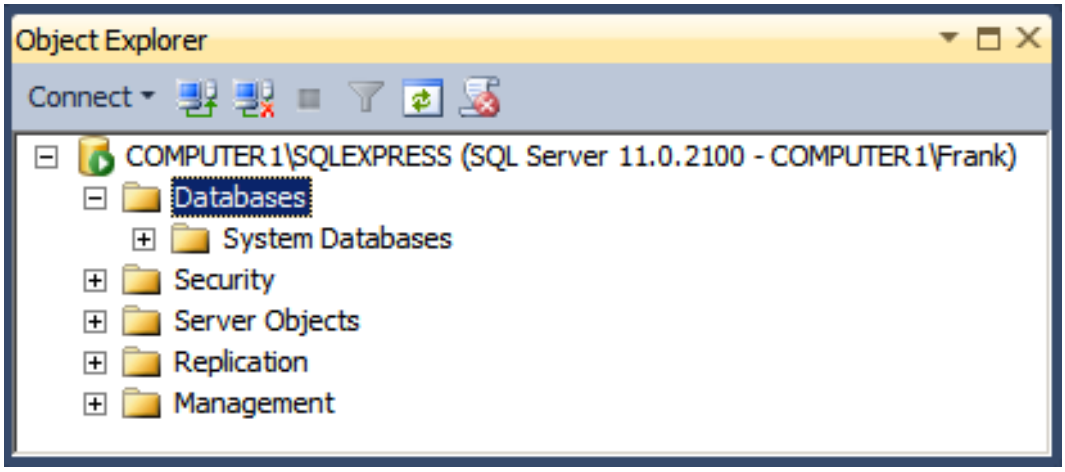
Relational Database Design & UML

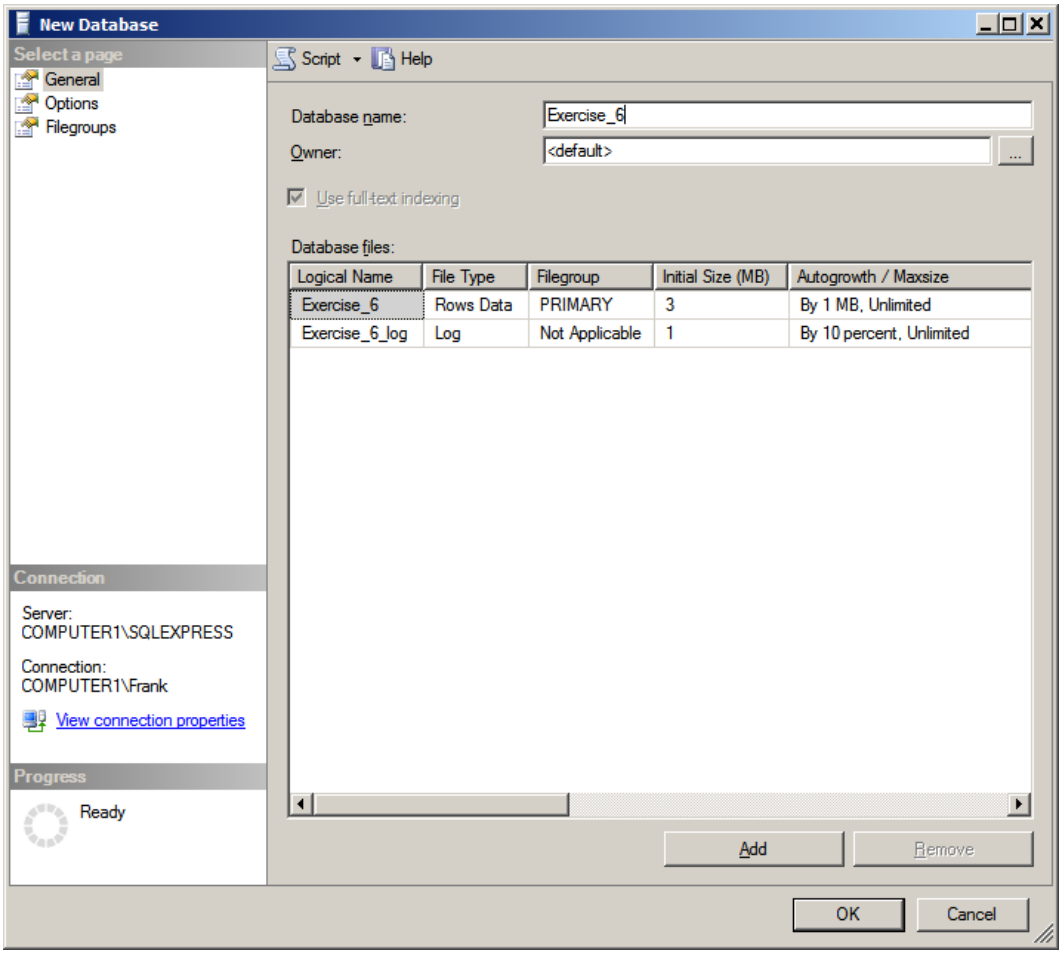
EXERCISE 6 – SQL Server Management Studio

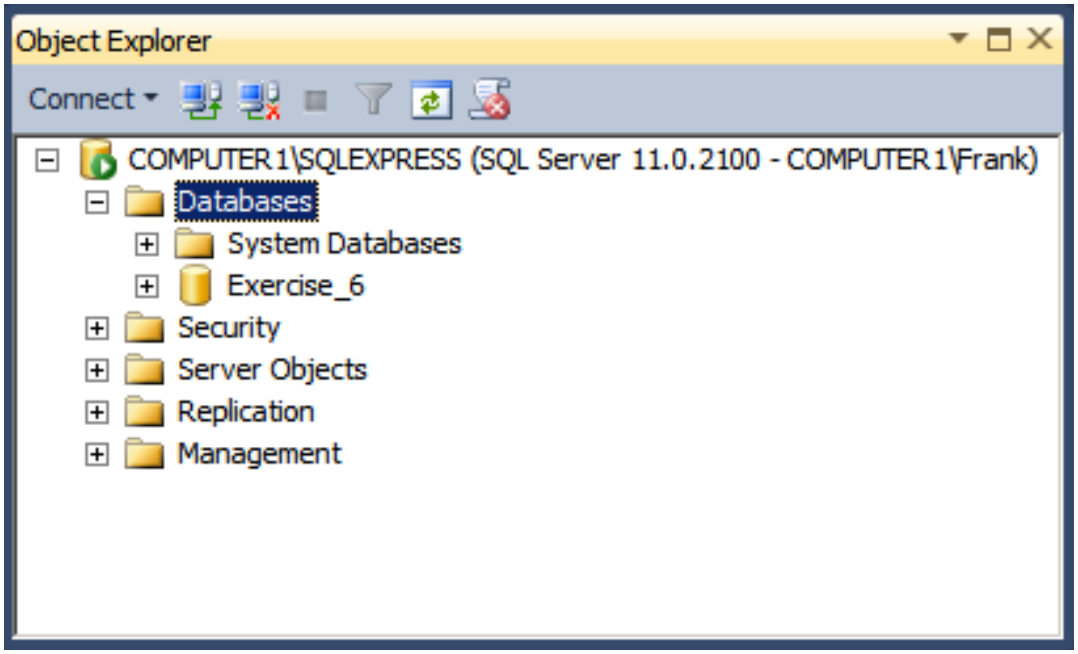
Complete the following using the Microsoft SQL Server Management Studio tool.

1. Launch and connect to SQL Server.

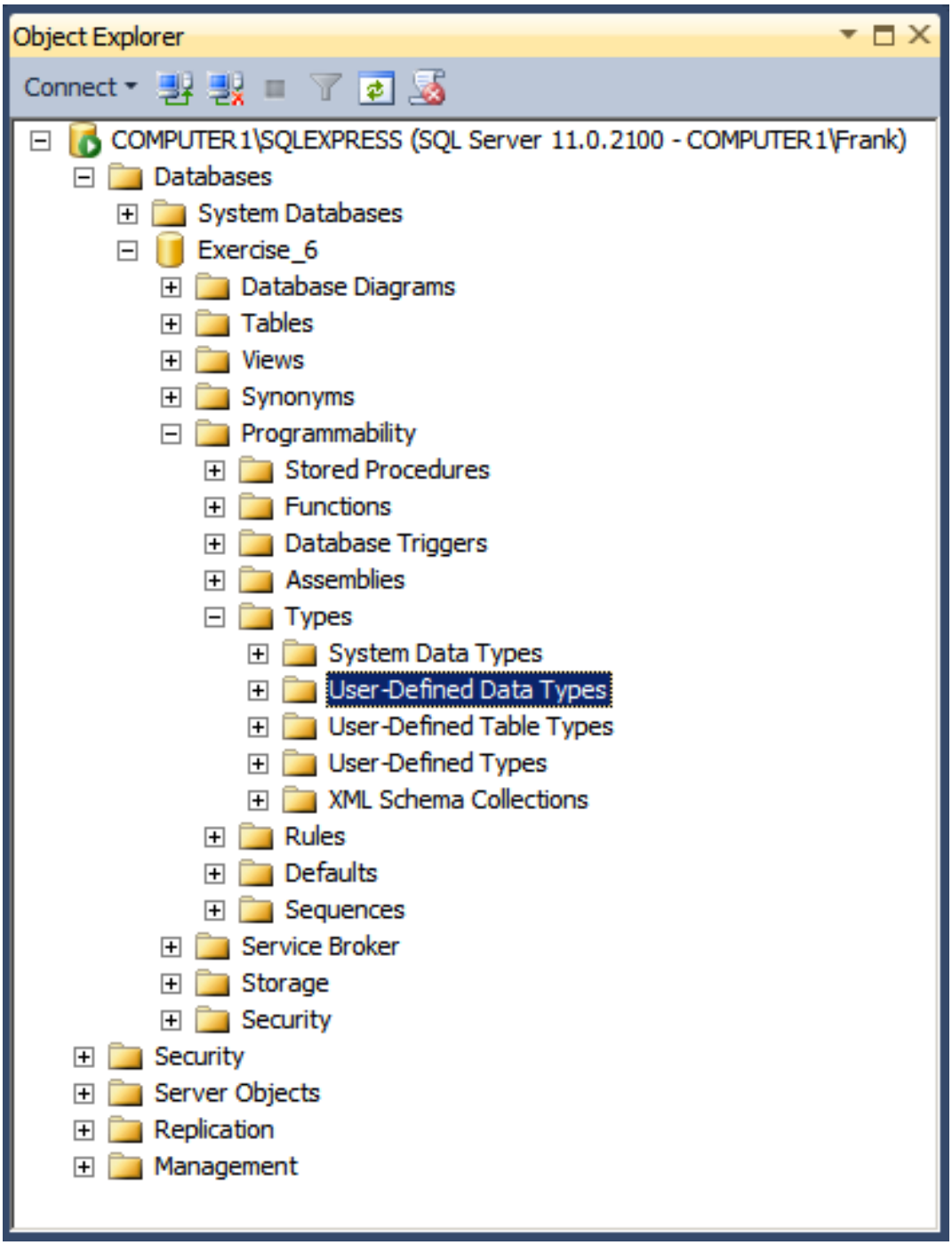
2. Create a database called **Exercise_6** (Right click).

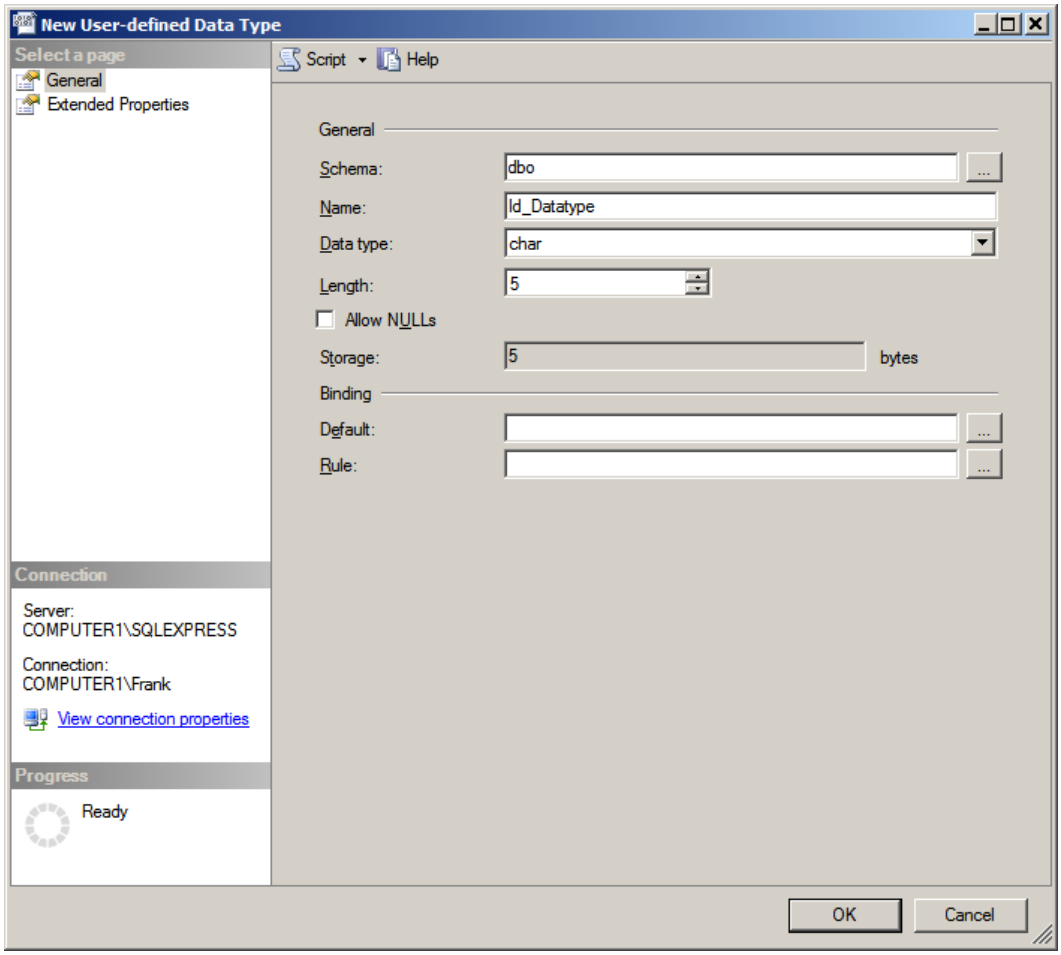


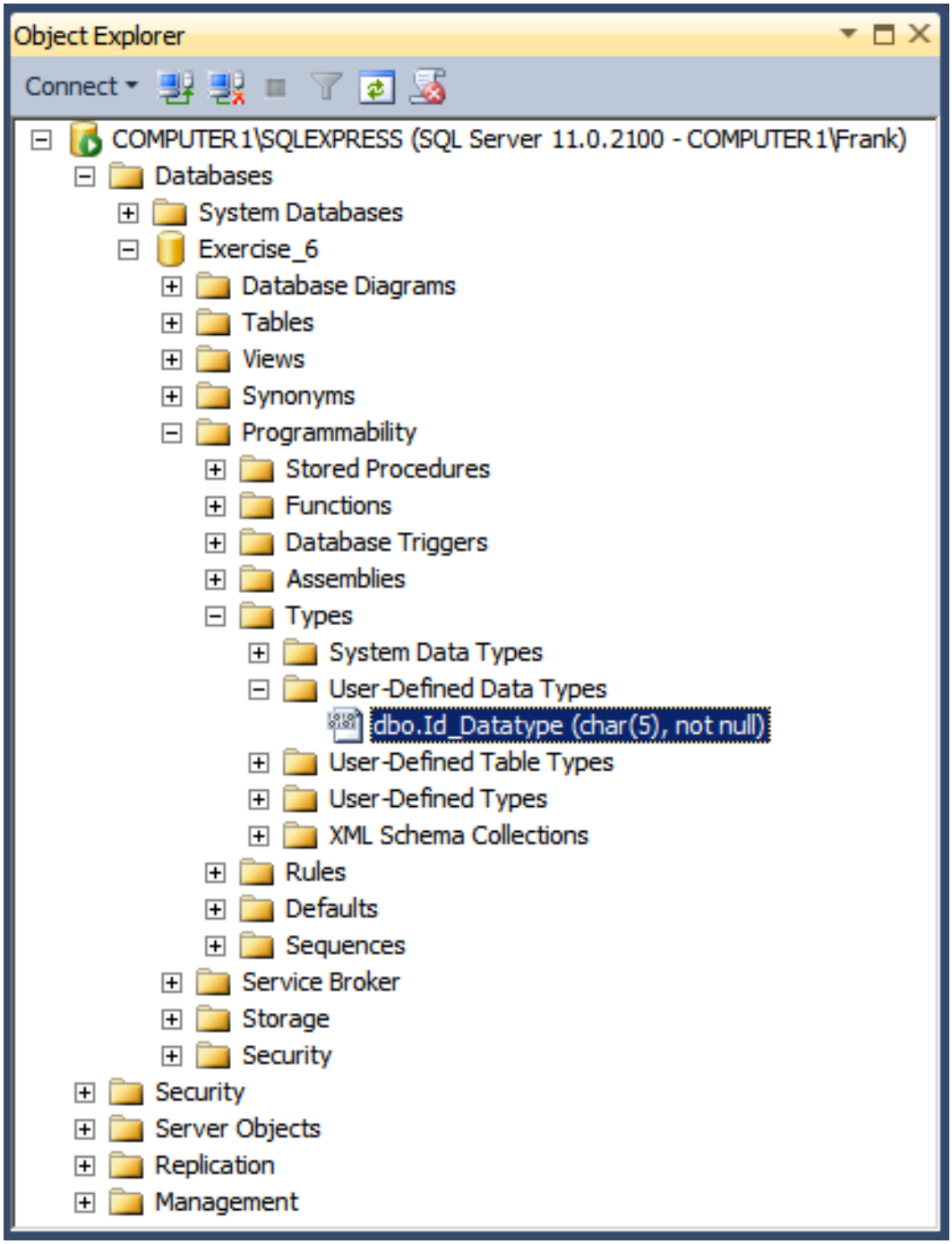



Object Explorer
Connect
COMPUTER1\SQLEXPRESS (SQL Server 11.0.2100 - COMPUTER1\Frank)
Databases
System Databases
Exercise_6
Security
Server Objects
Replication
Management

3. Create a user-defined data type called **Id_Datatype.** This will be used for the Id columns to ensure consistent data type, length, and null ability.

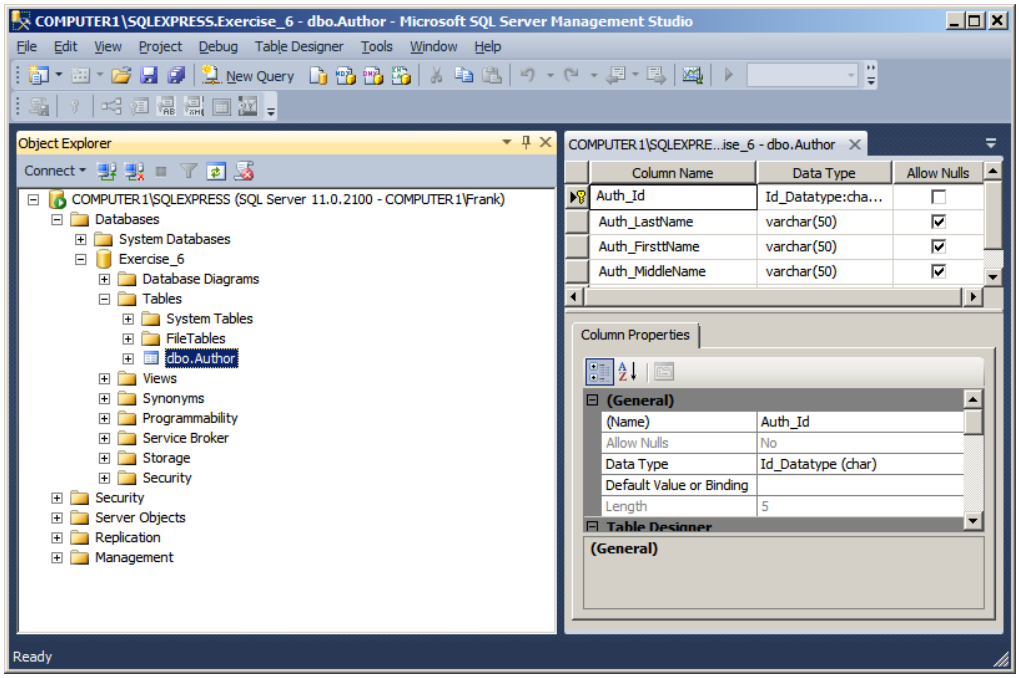



New User-defined Data Type
Select a page
General
Extended Properties
Script
Help
General
Schema:
dbo
Name:
Id_Datatype
Data type:
char
Length:
5
Allow NULLs
Storage:
5
bytes
Binding
Default:
Rule:
Connection
Server:
COMPUTER1\SQLEXPRESS
Connection:
COMPUTER1\Frank
View connection properties
Progress
Ready
OK
Cancel

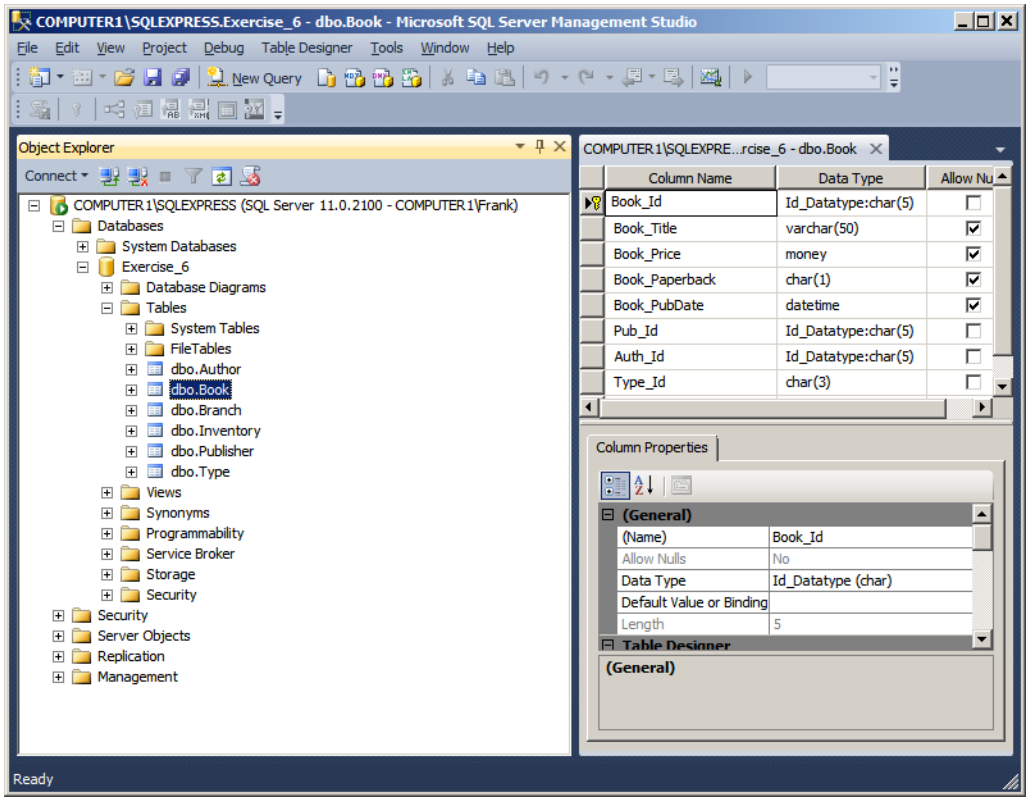

Object Explorer
Connect
COMPUTER1\SQLEXPRESS (SQL Server 11.0.2100 - COMPUTER1\Frank)
Databases
System Databases
Exercise_6
Database Diagrams
Tables
Views
Synonyms
Programmability
Stored Procedures
Functions
Database Triggers
Assemblies
Types
System Data Types
User-Defined Data Types
dbo.Id_Datatype (char(5), not null)
User-Defined Table Types
User-Defined Types
XML Schema Collections
Rules
Defaults
Sequences
Service Broker
Storage
Security
Security
Server Objects
Replication
Management

4. Create the following tables (see page 2 for column information):

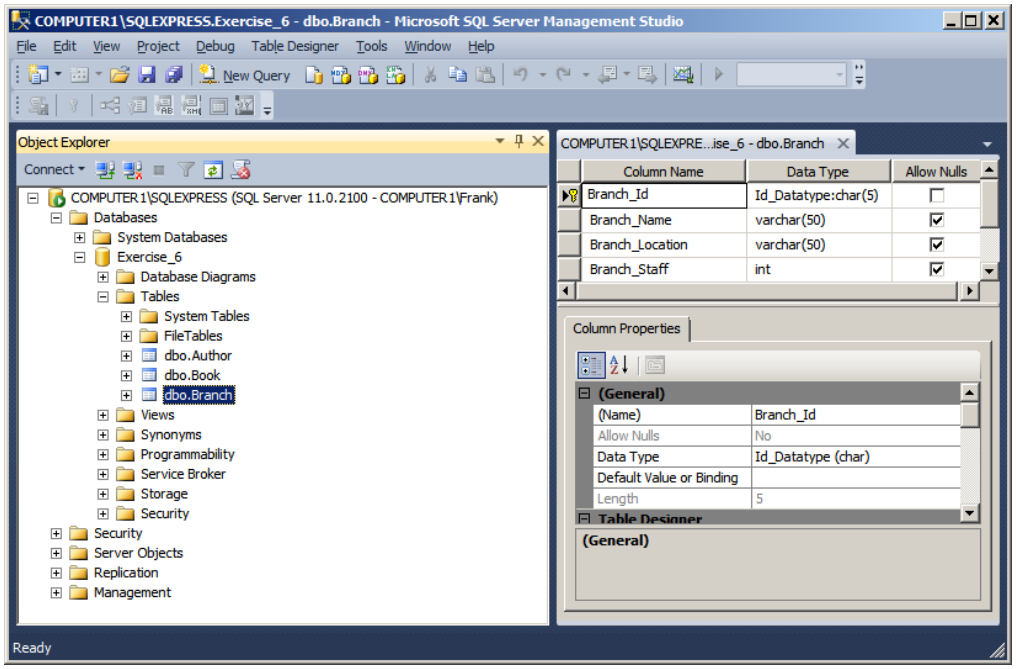
- Author



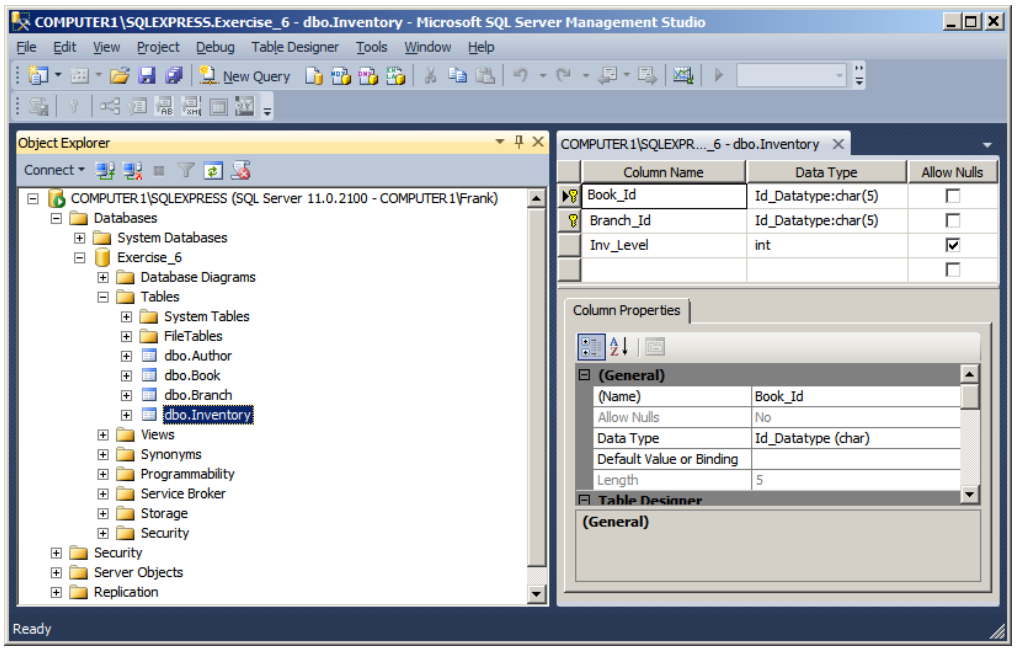
- Book



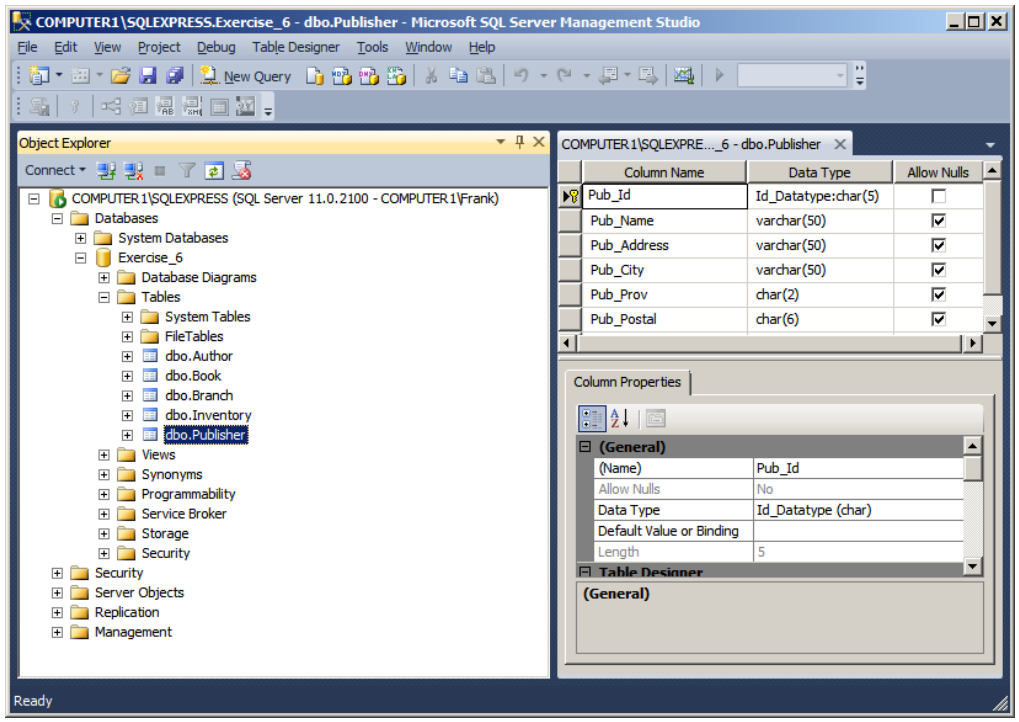
- Branch



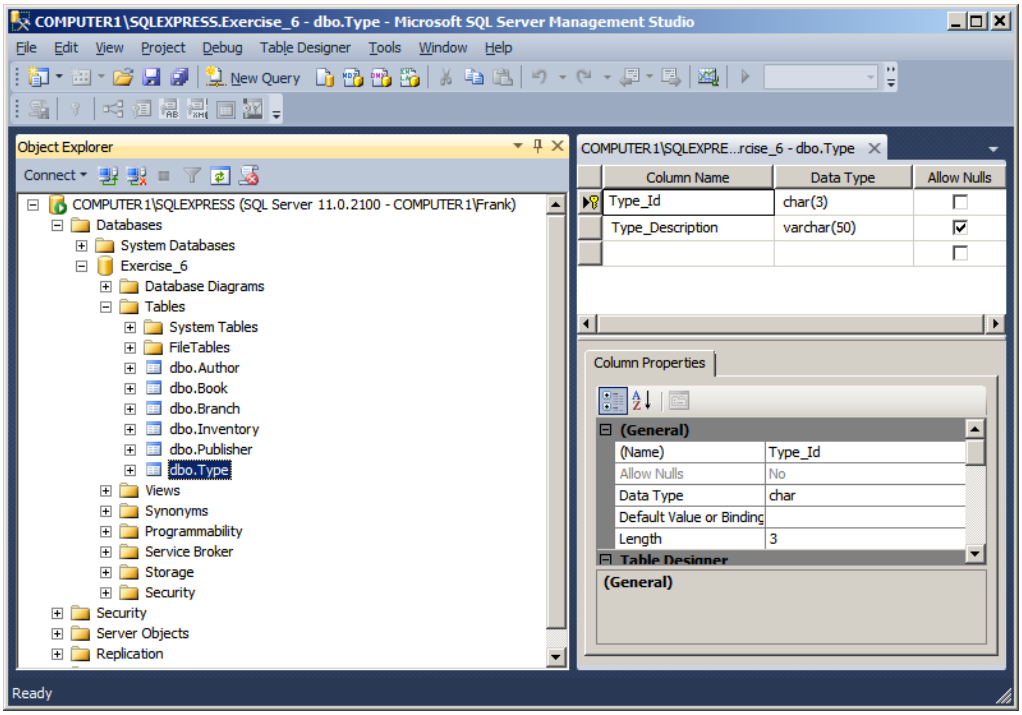
- Inventory



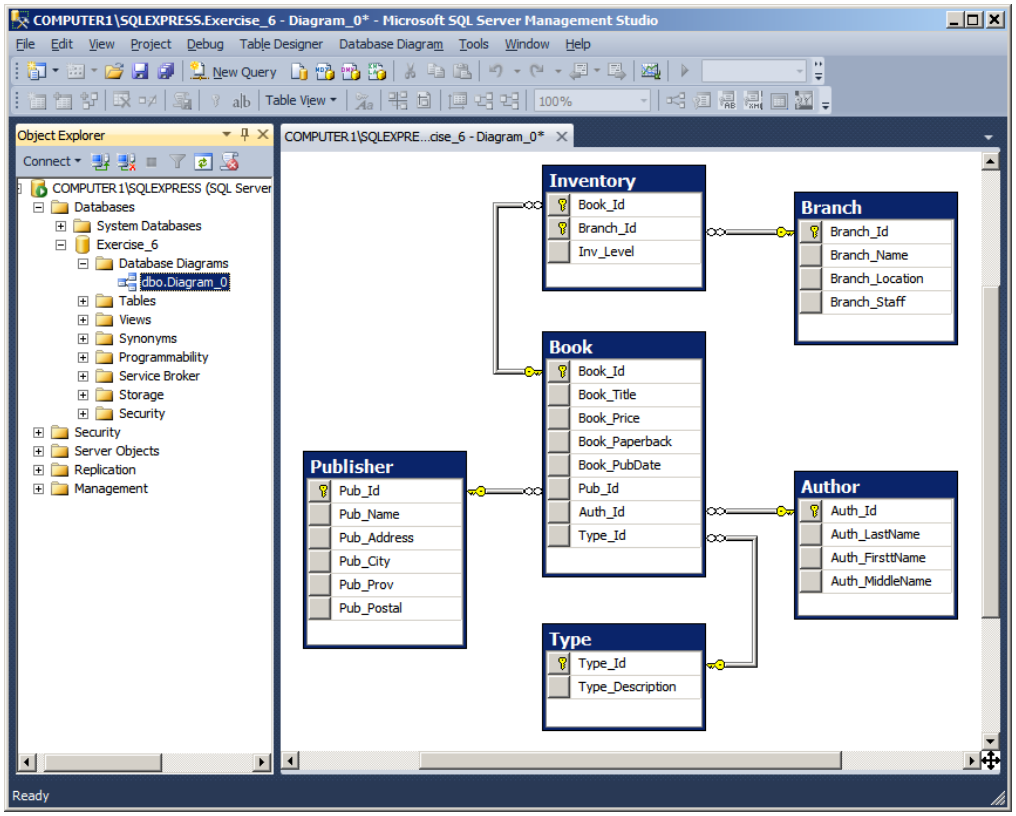
- Publisher



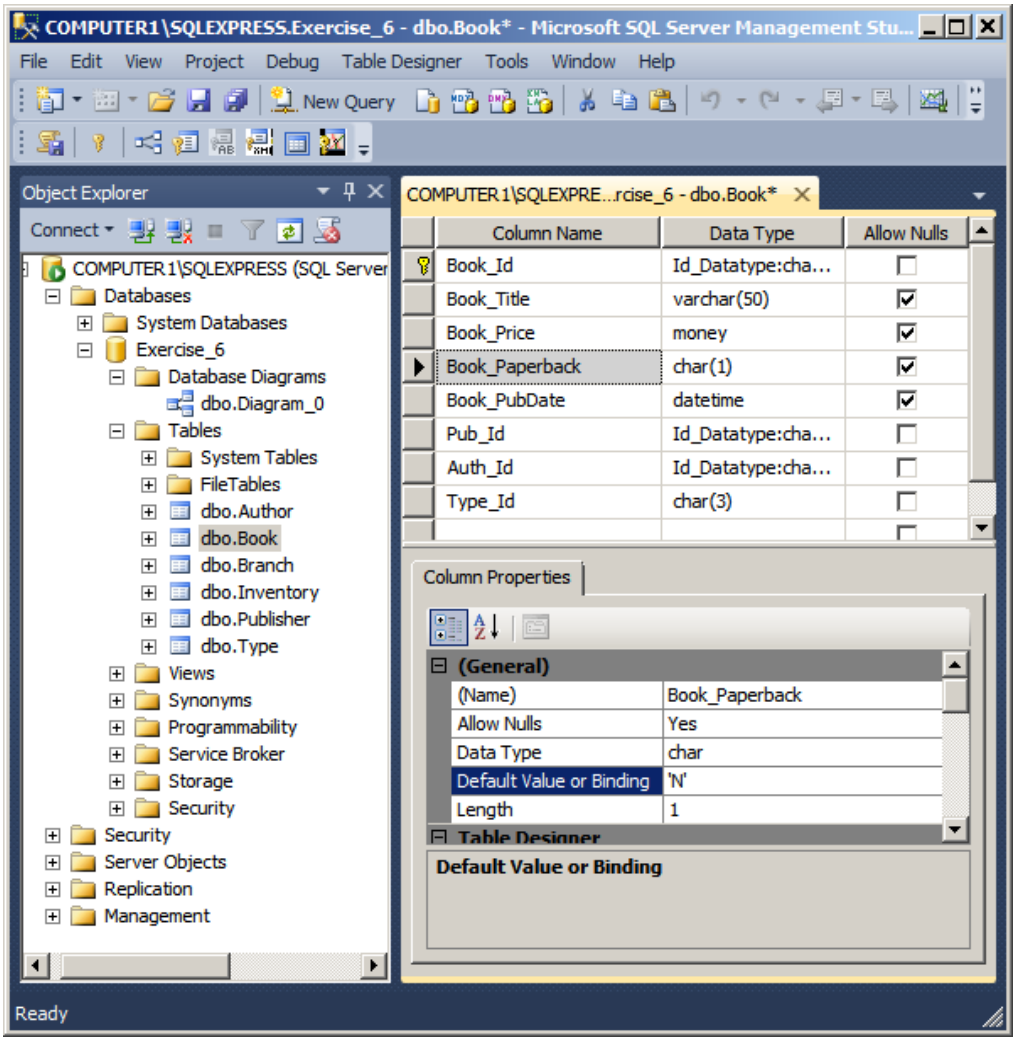
- Type



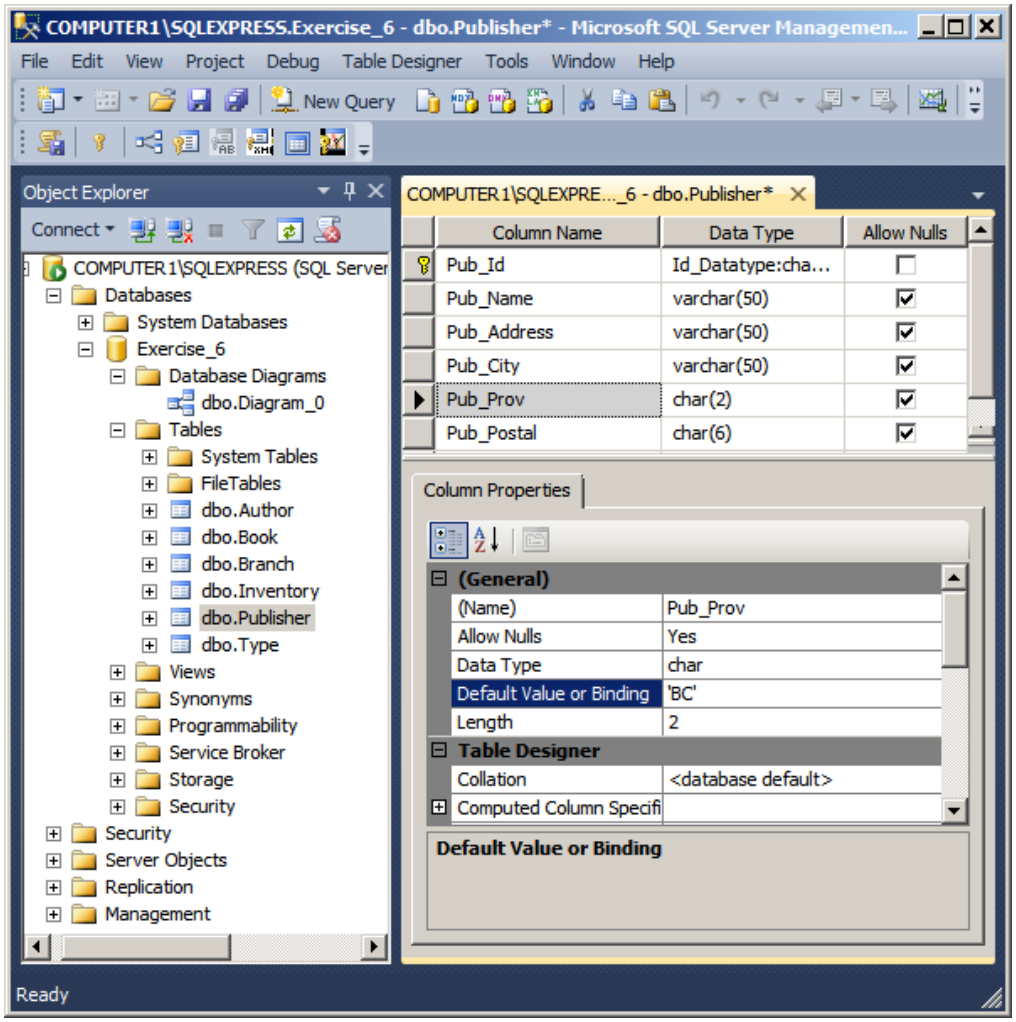
5. Add all the Primary Keys and Foreign Keys.



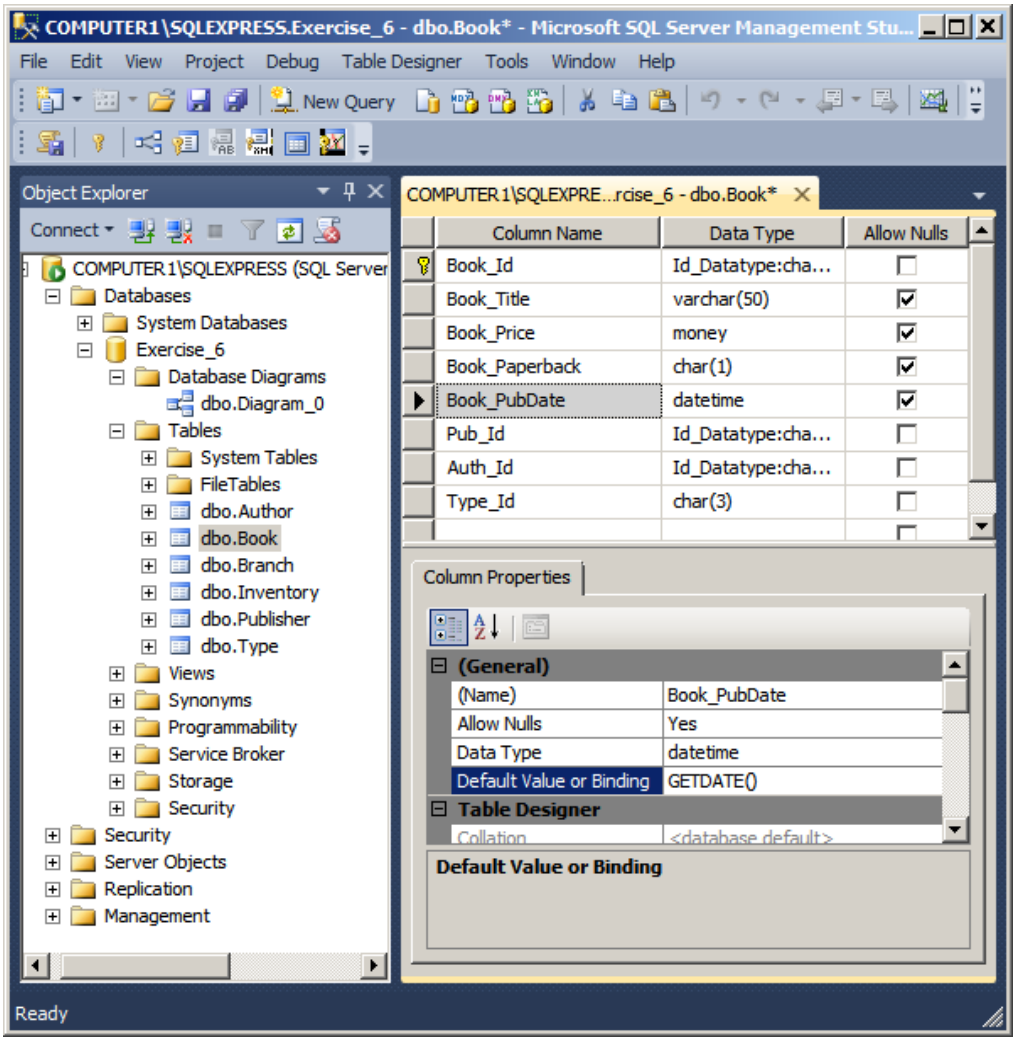
6. The default value for the **Book** table's **Paperback** column is **'N'**.



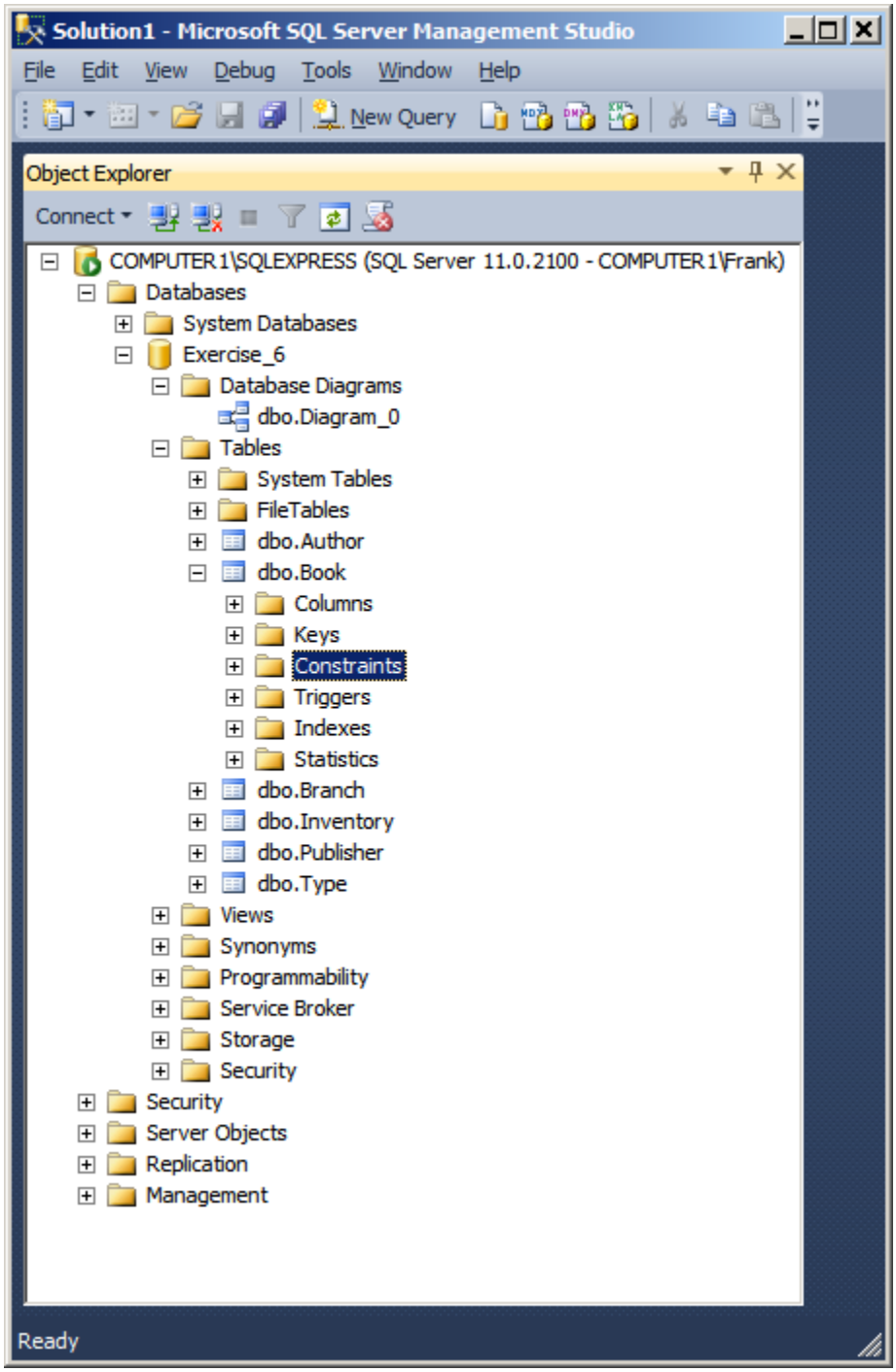
7. The default value for the **Publisher** table's **Prov** column is **'BC'**.

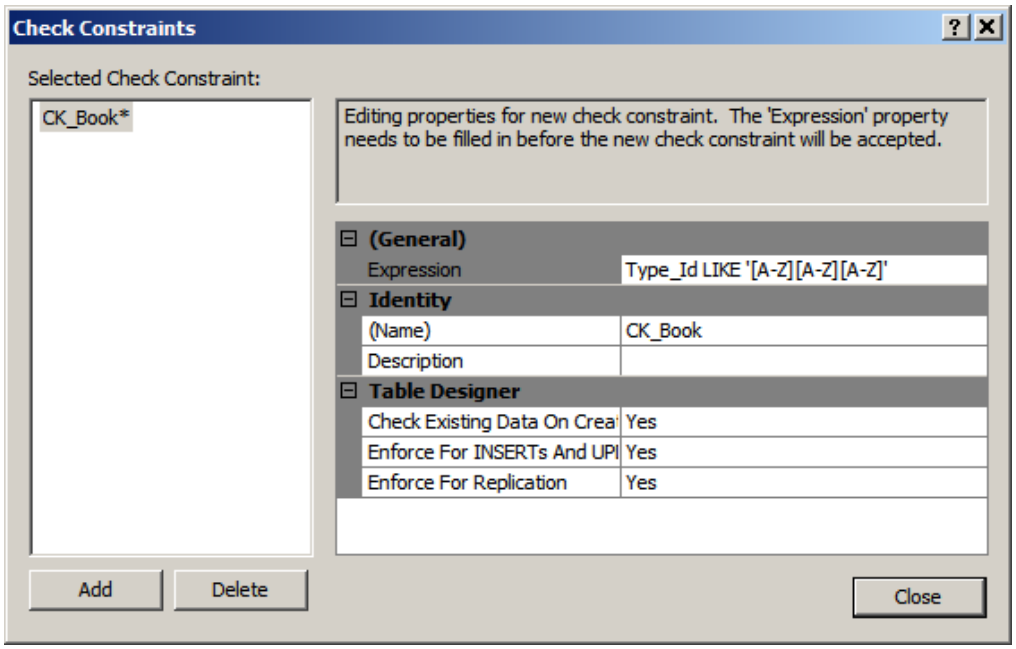


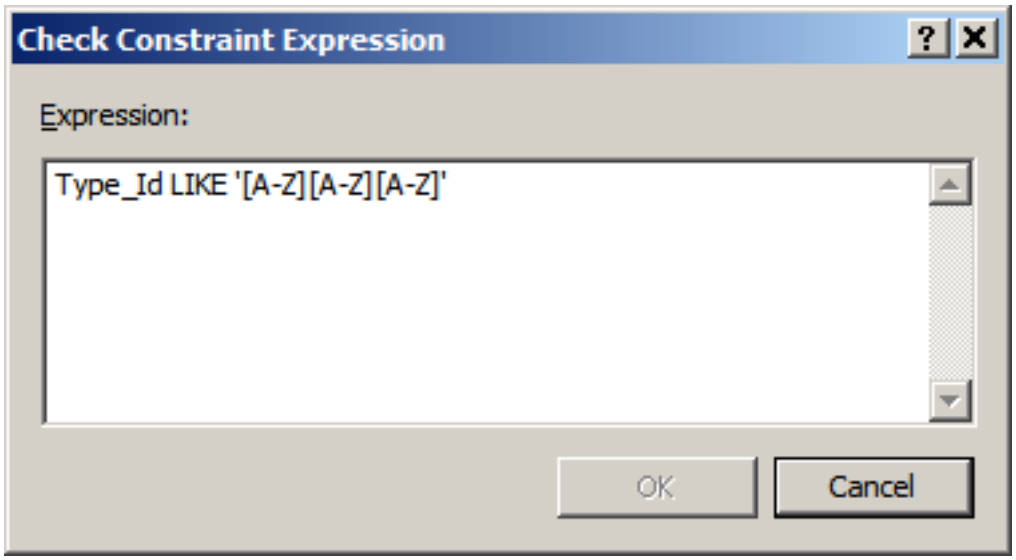
8. The default value for the **Book** table's **PubDate** column is today **GETDATE()**.



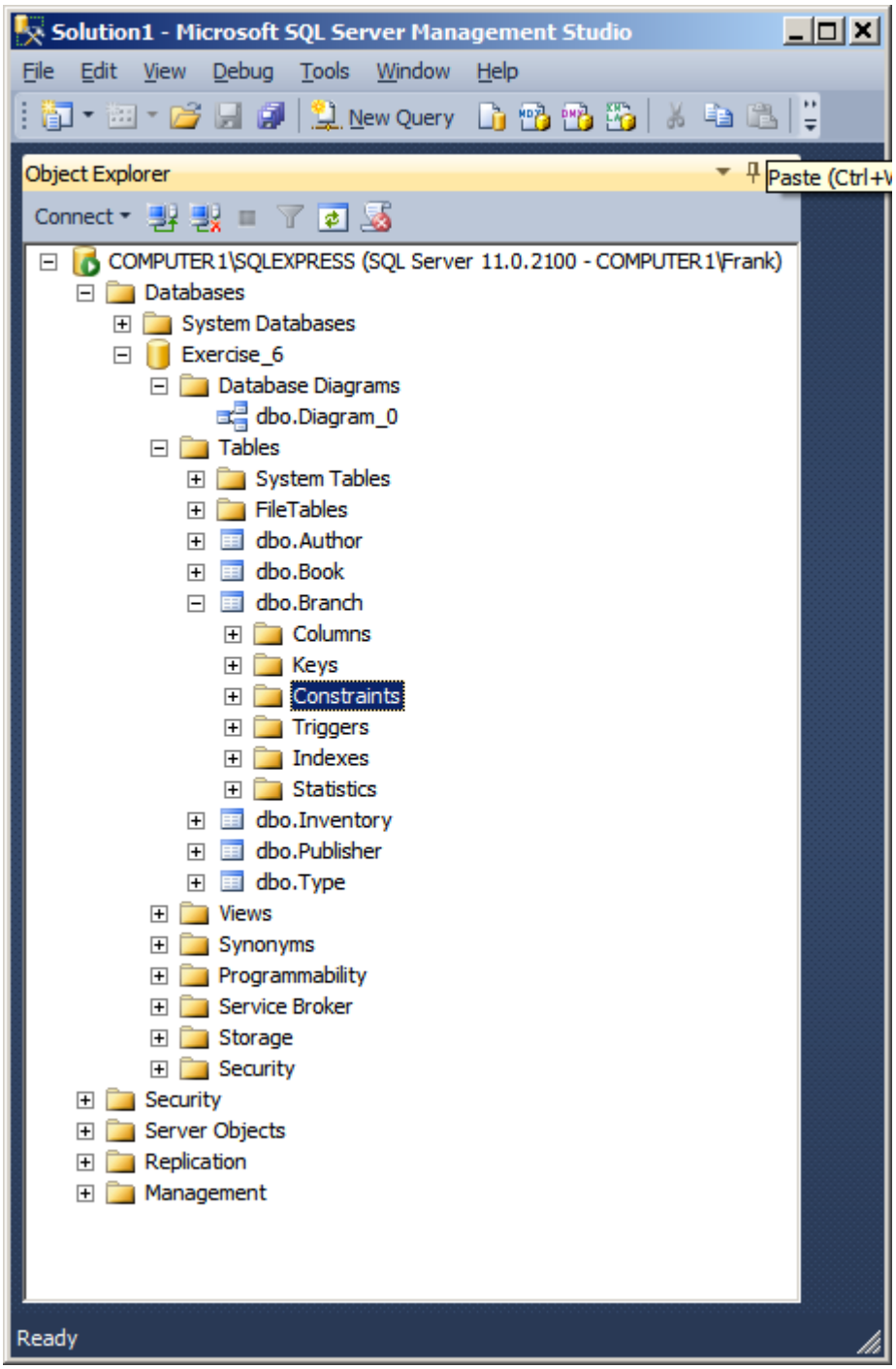
9. Add a check constraint to the **Book** table's **Type_Id** column to ensure the upper case letters A through Z are the only acceptable values for each of the characters in the column.

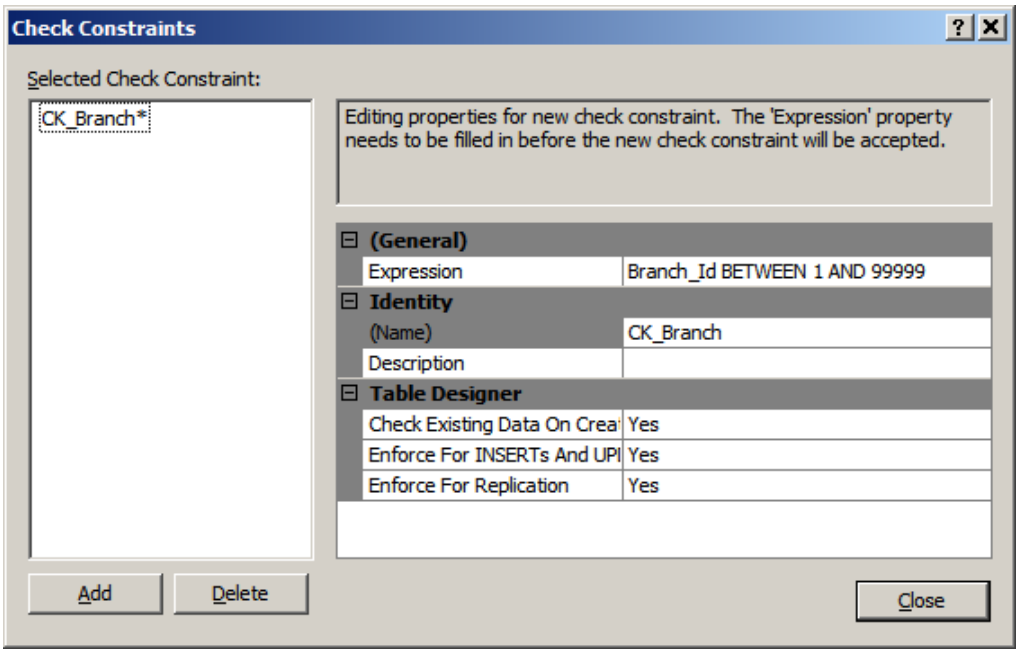


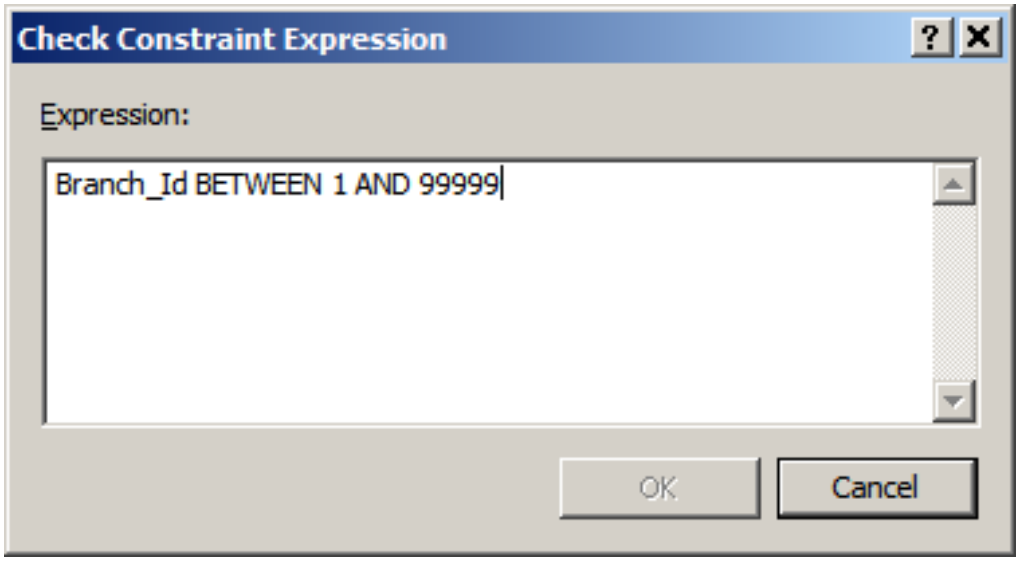

Check Constraints
Selected Check Constraint:
CK_Book*
Editing properties for new check constraint. The 'Expression' property needs to be filled in before the new check constraint will be accepted.
(General)
Expression
Type_Id LIKE '[A-Z][A-Z][A-Z]'
Identity
(Name)
CK_Book
Description
Table Designer
Check Existing Data On Crea
Yes
Enforce For INSERTs And UPI
Yes
Enforce For Replication
Yes
Add
Delete
Close


Check Constraint Expression
Expression:
Type_Id LIKE '[A-Z][A-Z][A-Z]'
OK
Cancel

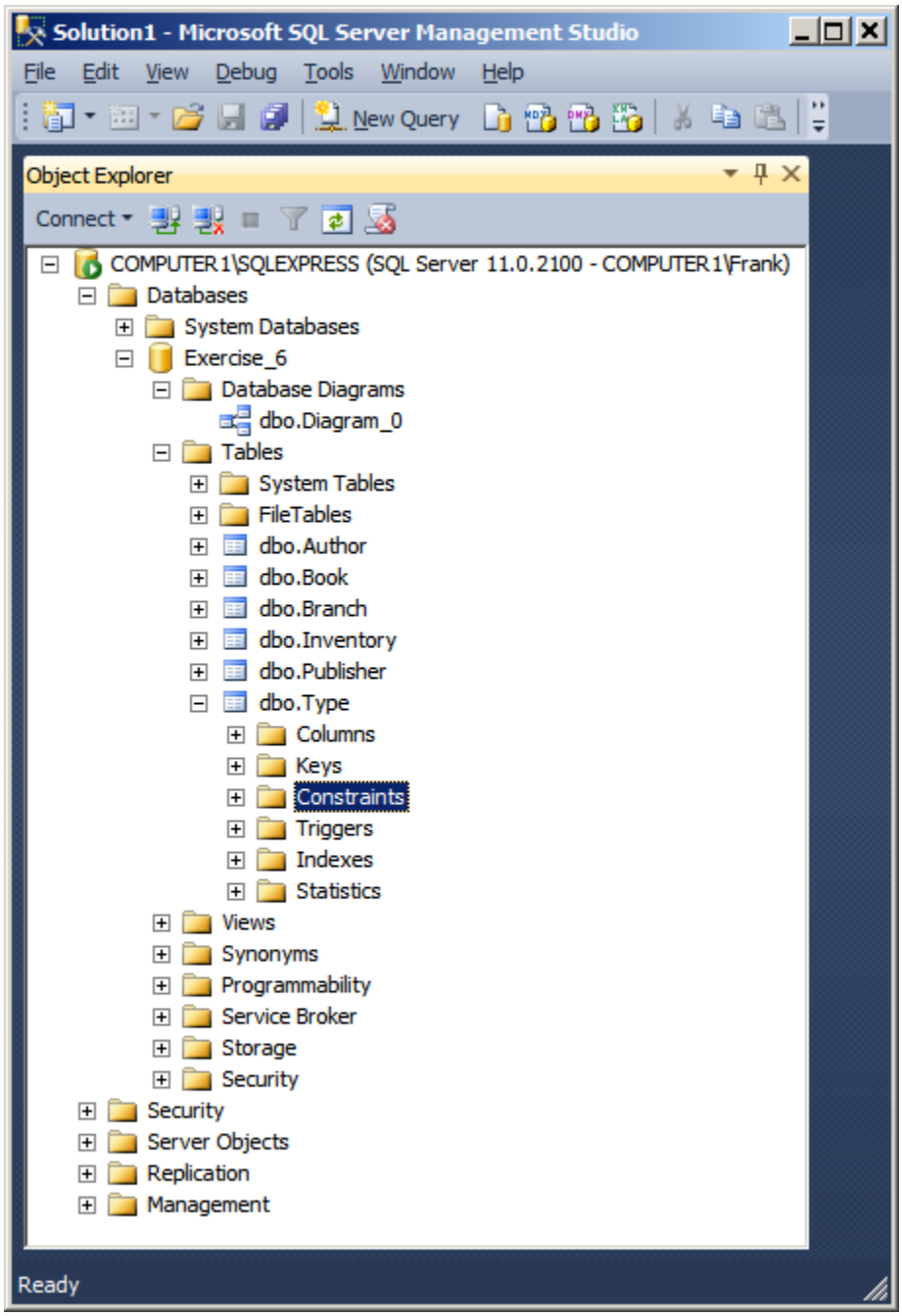
10. Add a check constraint to the **Branch** table's **Id** column for values 1 through 99999.

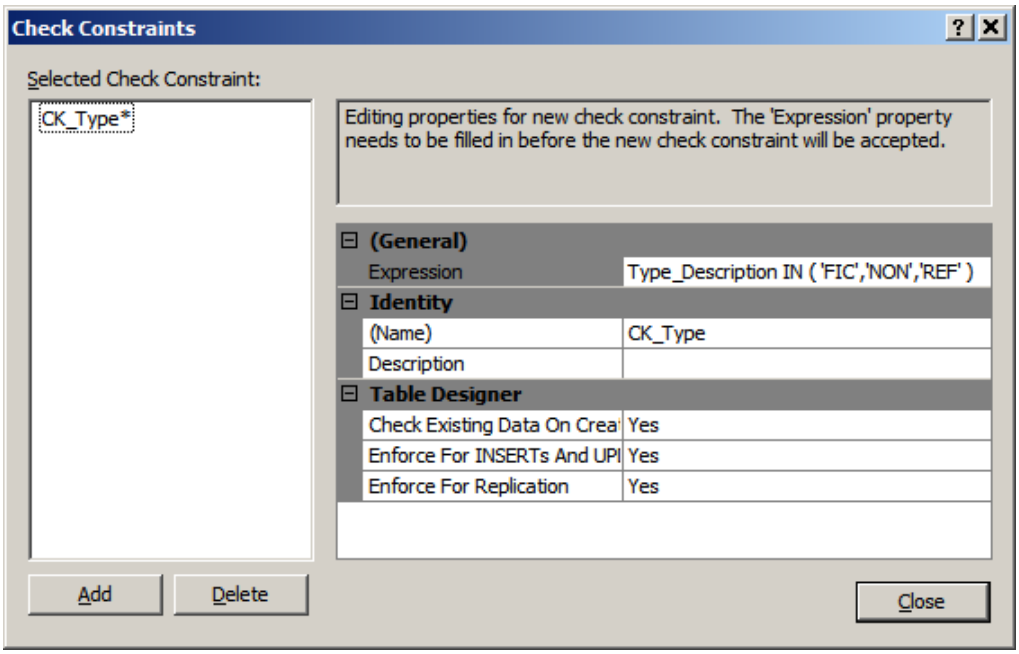


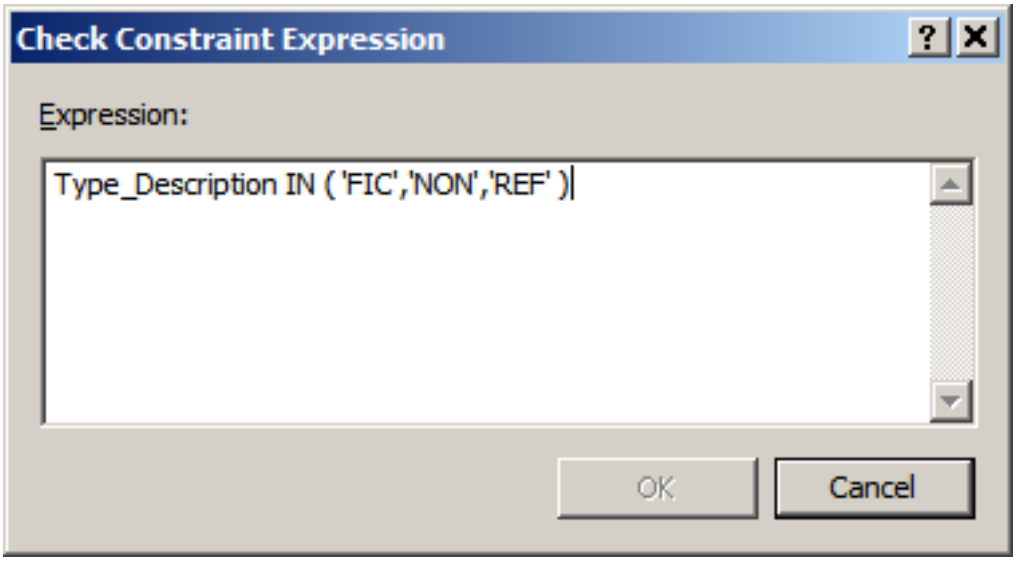

Check Constraints
Selected Check Constraint:
CK_Branch*
Editing properties for new check constraint. The 'Expression' property needs to be filled in before the new check constraint will be accepted.
(General)
Expression
Branch_Id BETWEEN 1 AND 99999
Identity
(Name)
CK_Branch
Description
Table Designer
Check Existing Data On Crea
Yes
Enforce For INSERTs And UPI
Yes
Enforce For Replication
Yes
Add
Delete
Close


Check Constraint Expression
Expression:
Branch_Id BETWEEN 1 AND 99999
OK
Cancel

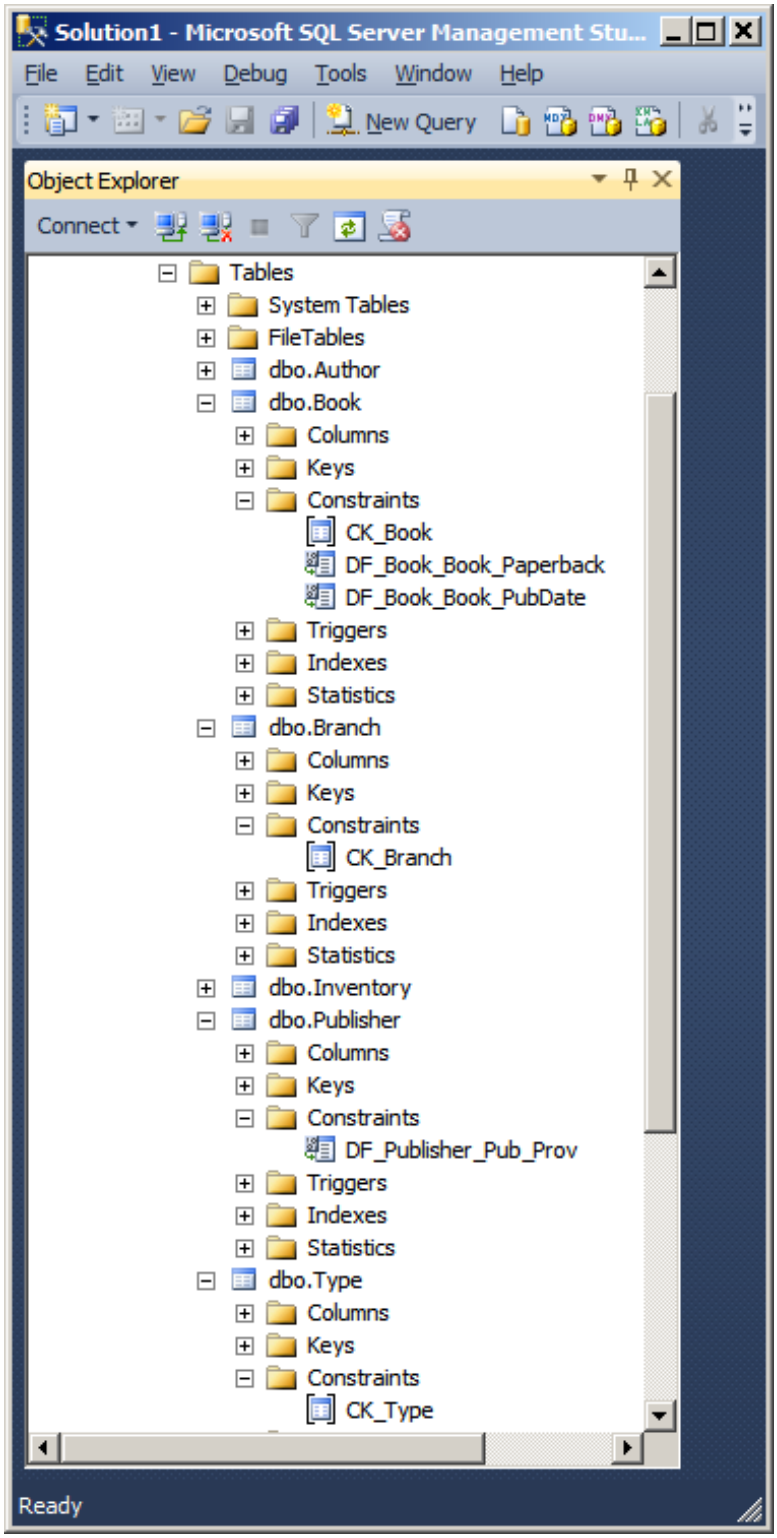
11. Add a check constraint to the **Type** table's **Description** column for the values 'FIC', 'NON', 'REF'.




Check Constraints
Selected Check Constraint:
CK_Type*
Editing properties for new check constraint. The 'Expression' property needs to be filled in before the new check constraint will be accepted.
(General)
Expression
Type_Description IN ('FIC','NON','REF')
Identity
(Name)
CK_Type
Description
Table Designer
Check Existing Data On Crea
Yes
Enforce For INSERTs And UPI
Yes
Enforce For Replication
Yes
Add
Delete
Close


Check Constraint Expression
Expression:
Type_Description IN ('FIC','NON','REF')
OK
Cancel

Summary of all constraints, including defaults.



Submit the PDF image of the ERD created in step 4 to the D2L Ex6 drop box.

Table Layout

Author

Column Name	Data Type	Length		Key	Null
Auth_Id	char	5	User-defined Data Type	PK	No
Auth_LastName	varchar	50			
Auth_FirstName	varchar	50			
Auth_MiddleName	varchar	50			

Book

Column Name	Data Type	Length		Key	Null
Book_Id	char	5	User-defined Data Type	PK	No
Book_Title	varchar	50			
Book_Price	money				
Book_Paperback	char	1			
Book_PubDate	datetime				
Pub_Id	char	5	User-defined Data Type	FK	No
Auth_Id	char	5	User-defined Data Type	FK	No
Type_Id	char	3		FK	No

Branch

Column Name	Data Type	Length		Key	Null
Branch_Id	char	5	User-defined Data Type	PK	No
Branch_Name	varchar	50			
Branch_Location	varchar	50			
Branch_Staff	int				

Inventory

Column Name	Data Type	Length		Key	Null
Book_Id	char	5	User-defined Data Type	PK1 FK	No
Branch_Id	char	5	User-defined Data Type	PK2 FK	No
Inv_Level	int				

Publisher

Column Name	Data Type	Length		Key	Null
Pub_Id	char	5	User-defined Data Type	PK	No
Pub_Name	varchar	50			
Pub_Address	varchar	50			
Pub_City	varchar	50			
Pub_Prov	char	2			
Pub_Postal	char	6			

Type

Column Name	Data Type	Length		Key	Null
Type_Id	char	3		PK	No
Type_Description	varchar	50			

Comp 1630

Relational Database Design & UML

EXERCISE 6 – SQL Server Management Studio

Complete the following using the Microsoft SQL Server Management Studio tool.

1. Launch and connect to SQL Server.

2. Create a database called **Exercise_6** (Right click).

```
CREATE DATABASE Exercise_6
ON (
    NAME = Exercise_6_data,
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL\Data\
    Exercise_6_data.mdf'
    )
LOG ON
    (
    NAME = Exercise_6_log,
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL\Data\
    Exercise_6_log.ldf'
    )
```

3. Create a user-defined data type called **Id_DataType.** This will be used for the Id columns to ensure consistent data type, length, and null ability.

```
EXEC sp_addtype  Id_Datatype, 'CHAR(5)', 'NOT NULL'
```

4. Create the following tables (see page 2 for column information):

   - Author

```
CREATE TABLE Author
(
        Auth_Id             Id_Datatype,
        Auth_LastName       VARCHAR(50),
        Auth_FirstName      VARCHAR(50),
        Auth_MiddleName     VARCHAR(50)
)
```

- Book

```
CREATE TABLE Book
(
        Book_Id             Id_Datatype,
        Book_Title          VARCHAR(50),
        Book_Price          MONEY,
        Book_Paperback      CHAR(1),
        Book_PubDate        DATETIME,
        Pub_Id              Id_Datatype,
        Auth_Id             Id_Datatype,
        Type_Id             CHAR(3)
)
```

- Branch

```
CREATE TABLE Branch
(
        Branch_Id           Id_Datatype,
        Branch_Name         VARCHAR(50),
        Branch_Location     VARCHAR(50),
        Branch_Staff        INT
)
```

- Inventory

```
CREATE TABLE Inventory
(
        Book_Id             Id_Datatype,
        Branch_Id           Id_Datatype,
        Inv_Level           INT
)
```

- Publisher

```
CREATE TABLE Publisher
(
        Pub_Id              Id_Datatype,
        Pub_Name            VARCHAR(50),
        Pub_Address         VARCHAR(50),
        Pub_City            VARCHAR(50),
        Pub_Prov            CHAR(2),
        Pub_postal          CHAR(6)
)
```

- Type

```
CREATE TABLE Type
(
        Type_Id             CHAR(3)  NOT NULL,
        Type_Description    VARCHAR(50)
)
```

5. Add all the Primary Keys and Foreign Keys.

```
ALTER TABLE Author
ADD PRIMARY KEY (Auth_Id)

ALTER TABLE Book
ADD PRIMARY KEY (Book_id)

ALTER TABLE Branch
ADD PRIMARY KEY (Branch_id)

ALTER TABLE Inventory
ADD PRIMARY KEY (Book_id, Branch_Id)

ALTER TABLE Publisher
ADD PRIMARY KEY (Pub_Id)

ALTER TABLE Type
ADD PRIMARY KEY (Type_Id)

ALTER TABLE Book
WITH CHECK ADD FOREIGN KEY (Pub_Id )
        REFERENCES Publisher (Pub_Id )

ALTER TABLE Book
WITH CHECK ADD FOREIGN KEY (Auth_Id)
        REFERENCES Author (Auth_Id)

ALTER TABLE Book
WITH CHECK ADD FOREIGN KEY (Type_Id)
        REFERENCES Type (Type_Id)
```

6. The default value for the **Book** table's **Paperback** column is **'N'**.

```
ALTER TABLE Book
ADD DEFAULT ( 'N' ) FOR Book_Paperback
```

7. The default value for the **Publisher** table's **Prov** column is **'BC'**.

```
ALTER TABLE Publisher
ADD DEFAULT ( 'BC' ) FOR Pub_Prov
```

8. The default value for the **Book** table's **PubDate** column is today **GETDATE()**.

```
ALTER TABLE Book
ADD DEFAULT ( GETDATE() ) FOR Book_PubDate
```

9. Add a check constraint to the **Book** table's **Type_Id** column to ensure the upper case letters A through Z are the only acceptable values for each of the characters in the column.

```
ALTER TABLE Book
ADD CONSTRAINT type_values CHECK ( Type_Id LIKE ( '[A-Z][A-Z][A-Z]') )
```

10. Add a check constraint to the **Branch** table's **Id** column for values 1 through 99999.

```
ALTER TABLE Branch
ADD CONSTRAINT branch_value CHECK ( Branch_Id BETWEEN 1 AND 99999 )
```

11. Add a check constraint to the **Type** table's **Description** column for the values 'FIC', 'NON', 'REF'.

```
ALTER TABLE Type
ADD CONSTRAINT booktype_values CHECK (Type_Description IN ( 'FIC','NON','REF' ) )
```

Submit the PDF image of the ERD created in step 4 to the D2L Ex6 drop box.

Table Layout

Author

Column Name	Data Type	Length		Key	Null
Auth_Id	char	5	User-defined Data Type	PK	No
Auth_LastName	varchar	50			
Auth_FirstName	varchar	50			
Auth_MiddleName	varchar	50			

Book

Column Name	Data Type	Length		Key	Null
Book_Id	char	5	User-defined Data Type	PK	No
Book_Title	varchar	50			
Book_Price	money				
Book_Paperback	char	1			
Book_PubDate	datetime				
Pub_Id	char	5	User-defined Data Type	FK	No
Auth_Id	char	5	User-defined Data Type	FK	No
Type_Id	char	3		FK	No

Branch

Column Name	Data Type	Length		Key	Null
Branch_Id	char	5	User-defined Data Type	PK	No
Branch_Name	varchar	50			
Branch_Location	varchar	50			
Branch_Staff	int				

Inventory

Column Name	Data Type	Length		Key	Null
Book_Id	char	5	User-defined Data Type	PK1 FK	No
Branch_Id	char	5	User-defined Data Type	PK2 FK	No
Inv_Level	int				

Publisher

Column Name	Data Type	Length		Key	Null
Pub_Id	char	5	User-defined Data Type	PK	No
Pub_Name	varchar	50			
Pub_Address	varchar	50			
Pub_City	varchar	50			
Pub_Prov	char	2			
Pub_Postal	char	6			

Type

Column Name	Data Type	Length		Key	Null
Type_Id	char	3		PK	No
Type_Description	varchar	50			

Comp 1630

Relational Database Design & UML

EXERCISE 7 - SQL

1. List the title ID, title, type, and price from the TITLES table where the notes column does not contain NULL values. Order the result set by title ID. The query should produce the result set listed below.

title_id	title	type	price
BU1032	The Busy Executive's Database Guide	business	19.99
BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	11.95
BU2075	You Can Combat Computer Stress!	business	2.99
BU7832	Straight Talk About Computers	business	19.99
MC2222	Silicon Valley Gastronomic Treats	mod_cook	19.99
...			
TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	20.95
TC4203	Fifty Years in Buckingham Palace Kitchens	trad_cook	11.95
TC7777	Sushi, Anyone?	trad_cook	14.99

(17 row(s) affected)

```
SELECT title_id,
       title,
       type,
       price
FROM   titles
WHERE  notes IS NOT NULL
ORDER BY title_id
```

2. List the DISTINCT store IDs from the SALES table whose order date is **September 14, 1994**. The query should produce the result set listed below.

stor_id
6380
7067
7131
8042

(4 row(s) affected)

```
SELECT DISTINCT (stor_id)
FROM   sales
WHERE  ord_date = 'Sep 14 1994'
```

3. List the name, phone number and states from the AUTHORS table. Format the name of the author as the first name followed by a space followed by the last name. Format the phone number as a left bracket followed by the area code followed by a right bracket followed by the phone number. Order the result set by the newly formatted name. The query should produce the result set listed below.

Name	Phone	State
Abraham Bennet	(415) 658-9932	CA
Akiko Yokomoto	(415) 935-4228	CA
Albert Ringer	(801) 826-0752	UT
Ann Dull	(415) 836-7128	CA
Anne Ringer	(801) 826-0752	UT
...		
Sheryl Hunter	(415) 836-7128	CA
Stearns MacFeather	(415) 354-7128	CA
Sylvia Panteley	(301) 946-8853	MD

(23 row(s) affected)

```
SELECT (au_fname + ' ' + au_lname)                              AS Name,
       '(' + SUBSTRING(phone,1,3) + ')' + SUBSTRING(phone,4,9) AS Phone,
       state                                                    AS State
FROM   authors
ORDER BY Name
```

4. List the title id, title, advance, year to date sales, and publishing date from the TITLES table where the advance is greater than or equal to **$3000**, and the publishing date is equal to **June 9, 1991 or June 12, 1991**. Display only the **first 30** characters of the title. Display the publisher date in the format of **DD-MM-YYYY**. Order the result set by the title. The query should produce the result set listed below.

TitleID	Title	Advance	YTDSales	PubDate
BU1111	Cooking with Computers: Surrep	5000.00	3876	09-06-1991
PS7777	Emotional Security: A New Algo	4000.00	3336	12-06-1991
TC4203	Fifty Years in Buckingham Pala	4000.00	15096	12-06-1991
TC7777	Sushi, Anyone?	8000.00	4095	12-06-1991
BU1032	The Busy Executive's Database	5000.00	4095	12-06-1991

(5 row(s) affected)

```
SELECT title_id                          AS TitleID,
       SUBSTRING(title,1,30)             AS Title,
       Advance                           AS Advance,
       ytd_sales                         AS YTDSales,
       CONVERT(CHAR(12),pubdate,105)     AS PubDate
FROM   titles
WHERE  advance >= 3000
  AND  (pubdate = 'Jun 9 1991' OR pubdate = 'Jun 12 1991')
ORDER BY Title
```

5. Find the **AVG** year-to-date sales, **MIN** year-to-date sales, and **MAX** year-to-date sales from the TITLES table. Rename the columns with an appropriate column heading. The query should produce the result set listed below.

AvgYTDSales	MinYTDSales	MaxYTDSales
6090	111	22246

(1 row(s) affected)

```
SELECT AVG(ytd_sales)      AS AvgYTDSales,
       MIN(ytd_sales)      AS MinYTDSales,
       MAX(ytd_sales)      AS MaxYTDSales
FROM   titles
WHERE  ytd_sales IS NOT NULL
```

6. List the authors from the AUTHORS table with a city of **Oakland**, **San Francisco,** or **Berkeley**, and a zip code of **94609**, **94130,** or **94705**. Order the result set by name. The query should produce the result set listed below.

AuthorID	Name	Phone	City	Zip
409-56-7008	Abraham Bennet	415 658-9932	Berkeley	94705
486-29-1786	Charlene Locksley	415 585-4620	San Francisco	94130
238-95-7766	Cheryl Carson	415 548-7723	Berkeley	94705
274-80-9391	Dean Straight	415 834-2919	Oakland	94609
724-08-9931	Dirk Stringer	415 843-2991	Oakland	94609
756-30-7391	Livia Karsen	415 534-9219	Oakland	94609

(6 row(s) affected)

```
SELECT au_id                              AS AuthorID,
      (au_fname + ' ' + au_lname)         AS Name,
       phone                              AS Phone,
       city                               AS City,
       zip                                AS Zip
FROM   authors
WHERE  city IN ('Oakland','San Francisco','Berkeley')
  AND  zip  IN ('94609','94130','94705')
ORDER BY Name
```

7. List the information from the ROYSCHED table where the royalty is greater than or equal to **15** and less than or equal to **20**. Order the result set by the title id. The query should produce the result set listed below.

title_id	lorange	hirange	royalty
BU1111	12001	16000	16
BU1111	16001	20000	18
BU1111	20001	24000	20
BU2075	5001	7000	16
…			
TC4203	16001	24000	16
TC4203	24001	32000	18
TC4203	32001	40000	20

(29 row(s) affected)

```
SELECT *
FROM   roysched
WHERE  royalty BETWEEN 15 AND 20
ORDER BY title_id
```

8. List the employee id, first name, last name, job id, and hire date from the EMPLOYEE table for employees with a last name beginning with the letter **A** or **S**, and a hire date greater than or equal to **January 1, 1990**. Display the hire date in the format of **MON DD YYYY**. Order the result set by the last name of the employee. The query should produce the result set listed below.

EmpID	FirstName	LastName	JobID	HireDate
PMA42628M	Paolo	Accorti	13	Aug 27 1992
PSA89086M	Pedro	Afonso	14	Dec 24 1990
VPA30890F	Victoria	Ashworth	6	Sep 13 1990
MMS49649F	Mary	Saveley	8	Jun 29 1993
CGS88322F	Carine	Schmitt	13	Jul 7 1992
MFS52347M	Martin	Sommer	10	Apr 13 1990

(6 row(s) affected)

```
SELECT emp_id                            AS EmpID,
       fname                             AS FirstName,
       lname                             AS LastName,
       job_id                            AS JobID,
       CONVERT(CHAR(12),hire_date,109)   AS HireDate
FROM   employee
WHERE  (lname LIKE 'A%' OR lname LIKE 'S%')
   AND hire_date >= 'Jan 1 1990'
ORDER BY lname
```

9. List the store id, order number, order date, and a new order date calculated by adding **10** days to the original order date from the SALES table where the quantity is less than or equal to **15**. Display the order date and new order date in the format of **YYYY.MM.DD**.

StoreID	OrderNumber	OrderDate	NewOrderDate
6380	6871	1994.09.14	1994.09.24
6380	722a	1994.09.13	1994.09.23
7067	D4482	1994.09.14	1994.09.24
7131	P3087a	1993.05.29	1993.06.08
7896	QQ2299	1993.10.28	1993.11.07
7896	TQ456	1993.12.12	1993.12.22
8042	423LL922	1994.09.14	1994.09.24
8042	423LL930	1994.09.14	1994.09.24

(8 row(s) affected)

```
SELECT stor_id                                          AS StoreID,
       ord_num                                          AS OrderNumber,
       CONVERT(CHAR(12),ord_date,102)                   AS OrderDate,
       CONVERT(CHAR(12),DATEADD(DAY,10,ord_date),102) AS NewOrderDate
FROM   sales
WHERE  qty <= 15
ORDER BY stor_id
```

10. List the employee id, first name, last name, the year of the hire date, and the number of years the employee has worked up to **January 1, 2008** from the EMPLOYEE table. Format the name as last name followed by a comma and space followed by the first name. The query should produce the result set listed below.

EmpId	Name	HireDate	Years
PMA42628M	Accorti, Paolo	1992	16
PSA89086M	Afonso, Pedro	1990	18
VPA30890F	Ashworth, Victoria	1990	18
H-B39728F	Bennett, Helen	1989	19
L-B31947F	Brown, Lesley	1991	17
...			
MFS52347M	Sommer, Martin	1990	18
GHT50241M	Thomas, Gary	1988	20
DBT39435M	Tonini, Daniel	1990	18

(43 row(s) affected)

```
SELECT emp_id                                    AS EmpID,
       (lname + ', ' + fname)                    AS Name,
        DATEPART(YEAR,hire_date)                 AS HireDate,
        DATEDIFF(YEAR,hire_date,'Jan 1 2008')    AS Years
FROM    employee
ORDER BY Name
```

Comp 1630

Relational Database Design & UML

EXERCISE 8 - SQL

1. List the title from the TITLES table, the order number and order date from the SALES table, and the store name from the STORES table. Display only the first **30** characters of the title. Display the order date in the format of **MMM DD YYYY**. There should be a row produced in the result set for each row in the titles table. Order the result set by the order number. The query should produce the result set listed below. (Hint: use **LEFT OUTER JOIN** statement)

Title	OrderNumber	OrderDate	StoreName
Net Etiquette	NULL	NULL	NULL
The Psychology of Computer Coo	NULL	NULL	NULL
The Gourmet Microwave	423LL922	Sep 14 1994	Bookbeat
The Busy Executive's Database	423LL930	Sep 14 1994	Bookbeat
The Busy Executive's Database	6871	Sep 14 1994	Eric the Read Books
.....			
Straight Talk About Computers	QQ2299	Oct 28 1993	Fricative Bookshop
Silicon Valley Gastronomic Tre	TQ456	Dec 12 1993	Fricative Bookshop
You Can Combat Computer Stress	X999	Feb 21 1993	Fricative Bookshop

(23 row(s) affected)

```
SELECT SUBSTRING(t.title,1,30)          AS Title,
       s.ord_num                        AS OrderNumber,
       CONVERT(CHAR(12),s.ord_date,109) AS OrderDate,
       st.stor_name                     AS StoreName
FROM                 titles t
LEFT OUTER JOIN      sales s      ON t.title_id = s.title_id
LEFT OUTER JOIN      stores st    ON s.stor_id = st.stor_id
ORDER BY s.ord_num
```

2. Create a new table called ‘business_books’ containing the title ID, title, price, publisher ID, and publish date columns, as well as the data, from the TITLES table for those rows which are of type ‘business’.

(4 row(s) affected)

```
SELECT title_id,
       title,
       price,
       pub_id,
       pubdate
INTO   business_books
FROM   titles
WHERE  type = 'business'
```

3. List the publisher name and the total of books by each title type. Display the publisher name from the PUBLISHERS table, the title type and MIN price from the TITLES table, and the SUM of the quantity from the SALES table. (Hint: Use a **GROUP BY** statement)

PublisherName	Type	MinPrice	Qty
Algodata Infosystems	business	11.95	55
Algodata Infosystems	popular_comp	20.00	80
Binnet & Hardley	mod_cook	2.99	50
Binnet & Hardley	psychology	21.59	20
Binnet & Hardley	trad_cook	11.95	80
New Moon Books	business	2.99	35
New Moon Books	psychology	7.00	173

(7 row(s) affected)

```
SELECT p.pub_name    AS PublisherName,
       t.type        AS Type,
       MIN(t.price)  AS MinPrice,
       SUM(s.qty)    AS Qty
FROM          titles t
INNER JOIN    sales s       ON t.title_id = s.title_id
INNER JOIN    publishers p  ON t.pub_id = p.pub_id
GROUP BY t.type, p.pub_name
```

4. Using the **UNION** command, calculate new prices for the books based on the year-to-date sales for each book in the TITLES table. If the year-to-date sales are less than $2500, add 15% to the price; if the year-to-date sales are greater than or equal to $2500 and less than or equal to $10000, add 10% to the price of the book; if the year-to-date sales are greater than $10000, add 5% to the price. Display the title id, year-to-date sales, price, and the new calculated price from the TITLES table. Order the result set by title id. The query should produce the result set listed below.

TitleID	YTDSales	Price	NewPrice
BU1032	4095	19.99	21.99
BU1111	3876	11.95	13.16
BU2075	18722	2.99	3.14
BU7832	4095	19.99	21.99
.....			
TC3218	375	20.95	24.09
TC4203	15096	11.95	12.55
TC7777	4095	14.99	16.49

(16 row(s) affected)

```
SELECT title_id                                AS TitleID,
       ytd_sales                               AS YTDSales,
       price                                   AS Price,
       CONVERT(DECIMAL(5,2),(price * 1.15))    AS NewPrice
FROM   titles
WHERE  ytd_sales < 2500
UNION
SELECT title_id,
       ytd_sales,
       price,
       CONVERT(DECIMAL(5,2),(price * 1.10))
FROM   titles
WHERE  ytd_sales BETWEEN 2500 AND 10000
```

```
UNION
SELECT title_id,
       ytd_sales,
       price,
       CONVERT(DECIMAL(5,2),(price * 1.05))
FROM   titles
WHERE  ytd_sales > 10000
ORDER BY title_id
```

5. List the AVG and SUM of the price by type for rows with a price that is NOT NULL from the TITLES table. At the end of the report, show the AVG and SUM of the price for all types. The query should produce the result set listed below. (Hint: Use the **GROUP BY WITH ROLLUP** statement)

Type	Average	Sum
business	13.73	54.92
mod_cook	11.49	22.98
popular_comp	21.475	42.95
psychology	13.504	67.52
trad_cook	15.9633	47.89
NULL	**14.7662**	**236.26**

(6 row(s) affected)

```
SELECT type          AS Type,
       AVG(price)    AS Average,
       SUM(price)    AS Sum
FROM   titles
WHERE  price IS NOT NULL
GROUP BY type WITH ROLLUP
```

6. For each unique store ID, list the store ID, store name, and SUM of the cost calculated as (quantity * price), but only for those stores with a cost between $500 and $1500. Obtain the store ID and name from the STORES table, the quantity from the SALES table, and the price from the TITLES table. Order the result set by store ID. The query should produce the result set listed below.

StoreID	StoreName	Cost
7067	News & Brews	1486.30
7131	Doc-U-Mat: Quality Laundry and Books	1400.15
7896	Fricative Bookshop	604.40
8042	Bookbeat	1232.00

(4 row(s) affected)

```
SELECT st.stor_id                  AS StoreID,
       st.stor_name                AS StoreName,
       SUM (s.qty * t.price)       AS Cost
FROM          stores st
INNER JOIN    sales s       ON st.stor_id = s.stor_id
INNER JOIN    titles t      ON s.title_id = t.title_id
GROUP BY      st.stor_id,
              st.stor_name
HAVING SUM (s.qty * t.price) BETWEEN 500.00 AND 1500.00
ORDER BY st.stor_id
```

7. For each store ID, list the SUM of the quantity from the SALES table and the MIN price from the TITLES table. Generate a final total of the qty SUM and MIN price. The query should produce the result set listed below.

StoreID	Qty	Min
6380	8	10.95
7066	125	10.95
7067	90	10.95
7131	130	2.99
7896	60	2.99
8042	80	2.99
NULL	**493**	**2.99**

(7 row(s) affected)

```
SELECT s.stor_id     AS StoreID,
       SUM(s.qty)    AS Qty,
       MIN(t.price)  AS Min
FROM          sales s
INNER JOIN    titles t  ON s.title_id = t.title_id
GROUP BY s.stor_id WITH ROLLUP
```

8. Using the INSERT INTO command, insert a new title into the TITLES table with a title ID of 'ZZ1234', a title of 'Microsoft SQL Server', a book type of 'computer', a publisher ID of '0877', a price of $89.99, and a publish date of 'September 29, 2008'. Check your results.

```
INSERT INTO titles
(      title_id,
       title,
       type,
       pub_id,
       price,
       pubdate       )
VALUES
(      'ZZ1234',
       'Microsoft SQL Server',
       'computer',
       '0877',
       89.99,
       'Sep 29 2008' )
```

9. Using the UPDATE command, increase the price by **10%** for the title created in question 8. Check your results.

```
UPDATE titles
SET price = (price * 1.10)
WHERE title_id = 'ZZ1234'
```

10. Delete the title created in question 8. Check your results.

```
DELETE FROM titles
WHERE title_id = 'ZZ1234'
```

Comp 1630
Relational Database Design & UML

EXERCISE 9 - SQL

1. Using **subqueries**, write a query to display the first name, last name, address, city, and state from the AUTHORS table, for authors who live in the state of 'CA', and have at least one book type of 'popular_comp' (found in the TITLES table via the TITLEAUTHOR table). Display the name of the author as last name, followed by a comma and a space, followed by the first name. Order the result set by the name of the author. The query should produce the result set listed below.

Name	Address	City	State
Carson, Cheryl	589 Darwin Ln.	Berkeley	CA
Dull, Ann	3410 Blonde St.	Palo Alto	CA
Hunter, Sheryl	3410 Blonde St.	Palo Alto	CA
Locksley, Charlene	18 Broadway Av.	San Francisco	CA

(4 row(s) affected)

```
SELECT (a.au_lname + ', ' + a.au_fname) AS Name,
       a.address                        AS Address,
       a.city                           AS City,
       a.state                          AS State
FROM   authors a
WHERE  a.state = 'CA'
  AND  a.au_id IN
       (SELECT ta.au_id
        FROM   titleauthor ta
        WHERE  ta.title_id IN
              (SELECT t.title_id
               FROM   titles t
               WHERE  t.type = 'popular_comp'))
ORDER BY Name
```

2. Rewrite the query in question 1 without using subqueries. The query should produce the same result set as in question 1.

```
SELECT DISTINCT
       (a.au_lname + ', ' + a.au_fname) AS Name,
       a.address                        AS Address,
       a.city                           AS City,
       a.state                          AS State
FROM          authors a
INNER JOIN    titleauthor ta ON a.au_id = ta.au_id
INNER JOIN    titles t       ON ta.title_id = t.title_id
WHERE a.state = 'CA'
  AND t.type  = 'popular_comp'
ORDER BY Name
```

3. Create a view called **vw_sales_title_info** to display the store ID, order date, and quantity from the SALES table, the store name from the STORES table, and the title, price, advance, and publish date from the TITLES table.

```
CREATE VIEW vw_sales_title_info
AS
SELECT s.stor_id,
       s.ord_date,
       s.qty,
       st.stor_name,
       t.title,
       t.price,
       t.advance,
       t.pubdate
FROM          sales s
INNER JOIN    stores st ON s.stor_id  = st.stor_id
INNER JOIN    titles t  ON s.title_id = t.title_id
```

4. Run the view **vw_sales_title_info** displaying the store ID, store name, title, and price where the price is equal to **$19.99**. Order the result set by the store ID. The view should produce the result set listed below.

StoreID	StoreName	Title	Price
6380	Eric the Read Books	The Busy Executive's Database Guide	19.99
7131	Doc-U-Mat: Quality Laundry and Books	Prolonged Data Deprivation: Four Case Studies	19.99
7896	Fricative Bookshop	Straight Talk About Computers	19.99
7896	Fricative Bookshop	Silicon Valley Gastronomic Treats	19.99
8042	Bookbeat	The Busy Executive's Database Guide	19.99

(5 row(s) affected)

```
SELECT stor_id       AS StoreID,
       stor_name     AS StoreName,
       title         AS Title,
       price         AS Price
FROM   vw_sales_title_info
WHERE  price = 19.99
ORDER BY stor_id
```

5. Create a view called **vw_insert_stores** to display the store ID, store name, and state from the STORES table.

```
CREATE VIEW vw_insert_stores
AS
SELECT stor_id,
       stor_name,
       state
FROM   stores
```

6. Using the view **vw_insert_stores**, insert a row into the STORES table with a store ID of **9999**, a store name of '**Peterson Books'**, and a state of '**UT'**. Check your results.

```
INSERT INTO vw_insert_stores
VALUES (9999,'Peterson Books','UT')
```

7. List the author ID, last name, city, state, and zip code from the AUTHORS table. The StateName column is generated using the values in the state column (for TN, IN, UT, and CA only). Check for zip codes that are less than 94300. Order the result set by the zip code. The query should produce the result set listed below. (Hint: Use the **CASE** command)

AuthorID	LastName	City	State	StateName	Zip
807-91-6654	Panteley	Rockville	MD	-	20853
527-72-3246	Greene	Nashville	TN	Tennessee	37215
722-51-5454	DeFrance	Gary	IN	Indiana	46403
712-45-1867	del Castillo	Ann Arbor	MI	-	48105
341-22-1782	Smith	Lawrence	KS	-	66044
899-46-2035	Ringer	Salt Lake City	UT	Utah	84152
998-72-3567	Ringer	Salt Lake City	UT	Utah	84152
172-32-1176	White	Menlo Park	CA	California	94025
486-29-1786	Locksley	San Francisco	CA	California	94130

(9 row(s) affected)

```
SELECT au_id         AS AuthorID,
       au_lname      AS LastName,
       city          AS City,
       state         AS State,
       StateName =
          CASE state
              WHEN 'CA' THEN 'California'
              WHEN 'TN' THEN 'Tennessee'
              WHEN 'IN' THEN 'Indiana'
              WHEN 'UT' THEN 'Utah'
              ELSE '-'
          END,
       zip           AS Zip
FROM   authors
WHERE  zip < '94300'
ORDER BY Zip
```

8. Using a **subquery**, list the publisher ID and name from the PUBLISHERS table, for those publishers who have published business books. The query should produce the result set listed below. (Hint: Use the **EXISTS** command)

PublisherID	Name
0736	New Moon Books
1389	Algodata Infosystems

(2 row(s) affected)

```
SELECT DISTINCT
       p.pub_id      AS PublisherID,
       p.pub_name    AS Name
FROM   publishers p
WHERE  EXISTS
      (SELECT *
       FROM  titles t
       WHERE t.pub_id = p.pub_id
         AND t.type = 'business')
```

9. Write the command to determine the index for the EMPLOYEE table.

```
sp_helpindex employee
```

10. Write the command to create a new composite index called **empinx** on the EMPLOYEE table for the columns **emp_id** and **hire_date**.

```
CREATE INDEX empinx
ON employee(emp_id,hire_date)
```

Comp 1630
Relational Database Design & UML

EXERCISE 10 - SQL

1. List the first and last name of the employee having a last name beginning with **Thomas**. Use local variables for the first and last name and the **@@ROWCOUNT** command. Display the first and last name if the name is found, and the message 'Employee not found' if the last name does not exist in the EMPLOYEE table. The query should produce the result set listed below.

   Employee Name is Gary Thomas

```
DECLARE @lname varchar(30)
DECLARE @fname varchar(30)
SET    @lname = ' '
SET    @fname = ' '
SELECT @lname = lname,
       @fname = fname
FROM   employee
WHERE  lname LIKE 'Thomas%'
IF @@ROWCOUNT > 0
   PRINT 'Employee Name is ' + @fname + ' ' + @lname
ELSE
   PRINT 'Employee not found'
```

2. List employees with a hire date between **January 1 1989** and **December 31 1990**. Display the employee ID, first name, last name, hire date, and job ID from the EMPLOYEE table, and the job description from the JOB table. Use local variables for the two dates. Display the name of the employee as the last name, followed by a comma and a space, followed by the first name. Display the hire date in the format of **MMM DD YYYY**. Order the result set by the employee name. The query should produce the result set listed below.

EmployeeID	Name	HireDate	JobID	Description
---	---	---	---	---
PSA89086M	Afonso, Pedro	Dec 24 1990	14	Designer
VPA30890F	Ashworth, Victoria	Sep 13 1990	6	Managing Editor
H-B39728F	Bennett, Helen	Sep 21 1989	12	Editor
.....				
A-R89858F	Roulet, Annette	Feb 21 1990	6	Managing Editor
MFS52347M	Sommer, Martin	Apr 13 1990	10	Productions Manager
DBT39435M	Tonini, Daniel	Jan 1 1990	11	Operations Manager

   (15 row(s) affected)

```
DECLARE @hire_date1 datetime
DECLARE @hire_date2 datetime
SET    @hire_date1 = 'Jan 1 1989'
SET    @hire_date2 = 'Dec 31 1990'
SELECT e.emp_id                          AS EmployeeID,
       (e.lname + ', ' + e.fname)        AS Name,
       CONVERT(char(12),e.hire_date,109) AS HireDate,
       e.job_id                          AS JobID,
       j.job_desc                        AS Description
FROM         employee e
INNER JOIN   jobs j       ON e.job_id = j.job_id
WHERE e.hire_date BETWEEN @hire_date1 AND @hire_date2
ORDER BY Name
```

3. Create a stored procedure called **author_information** which takes **2 input parameters** consisting of an author ID and a title ID, and returns **3 output parameters** consisting of the last name, first name, and royalty percentage. If the author ID and title ID matches the input parameters, the stored procedure should return the last and first names from the AUTHORS table, and the royalty percentage from the TITLEAUTHOR table.

```
CREATE PROCEDURE author_information
(      @au_id       varchar(11),
       @title_id    varchar(6),
       @last_name   varchar(40)   OUTPUT,
       @first_name  varchar(20)   OUTPUT,
       @royaltyper  int           OUTPUT )
AS
SELECT @last_name = a.au_lname,
       @first_name = a.au_fname,
       @royaltyper = ta.royaltyper
FROM          authors a
INNER JOIN    titleauthor ta ON a.au_id = ta.au_id
WHERE a.au_id     = @au_id
  AND ta.title_id = @title_id
GO
```

4. Run the stored procedure **author_information** for author ID '**672-71-3249'** and the title ID '**TC7777'**. Display the output values from the stored procedure for the first name, last name, and royalty percentage. The stored procedure should produce the result below.

Author: Akiko Yokomoto
Royalty percentage = 40

```
DECLARE @last_name  varchar(40)
DECLARE @first_name varchar(40)
DECLARE @royaltyper int
EXEC   author_information '672-71-3249','TC7777',
                          @last_name   OUTPUT,
                          @first_name  OUTPUT,
                          @royaltyper  OUTPUT
PRINT  'Author: ' + @first_name + ' ' + @last_name
PRINT  'Royalty percentage = ' + CONVERT(char(20),@royaltyper)
```

5. Create a stored procedure called **store_information** which takes an **input variable** for the price of a book. The stored procedure should list the store ID and order date from the SALES table, the store name from the STORES table, and the title id, price, and advance from the TITLES table, where the price is greater than or equal to the input variable. Display the order date in the format of **YYYY.MM.DD**. Order the result set by the store ID.

```
CREATE PROCEDURE store_information
(      @price   money      )
AS
SELECT s.stor_id                        AS StoreID,
       st.stor_name                     AS Name,
       CONVERT(CHAR(12),s.ord_date,102) AS OrderDate,
       t.title_id                       AS TitleID,
       t.price                          AS Price,
       t.advance                        AS Advance
FROM          sales s
INNER JOIN    stores st     ON s.stor_id  = st.stor_id
INNER JOIN    titles t      ON s.title_id = t.title_id
WHERE t.price >= @price
ORDER BY s.stor_id
GO
```

6. Run the stored procedure **store_information** using a value of **$15.00** for the price. The stored procedure should produce the result set listed below.

Store_ID	Name	OrderDate	TitleID	Price	Advance
6380	Eric the Read Books	1994.09.14	BU1032	19.99	5000.00
7066	Barnum's	1993.05.24	PC8888	20.00	8000.00
7067	News & Brews	1992.06.15	TC3218	20.95	7000.00
7131	Doc-U-Mat: Quality Laundry and Books	1993.05.29	PS1372	21.59	7000.00
7131	Doc-U-Mat: Quality Laundry and Books	1993.05.29	PS3333	19.99	2000.00
7896	Fricative Bookshop	1993.10.28	BU7832	19.99	5000.00
7896	Fricative Bookshop	1993.12.12	MC2222	19.99	.00
8042	Bookbeat	1994.09.14	BU1032	19.99	5000.00
8042	Bookbeat	1993.05.22	PC1035	22.95	7000.00

(9 row(s) affected)

```
EXEC store_information 15.00
```

7. Create an **INSERT** trigger attached to the SALES table called **tr_insert_ytd**. The trigger should add the quantity inserted into the SALES table to the ytd sales column in the TITLES table (Hint: use UPDATE). Use the following code to test your trigger and query the TITLES table before and after to ensure that the ytd sales has, in fact, been increased by 5 for title ID 'PS7777'.

INSERT sales
VALUES ('7131', 'Q789', 'Mar 1 2007', 5, 'Net 30', 'PS7777')

```
CREATE TRIGGER tr_insert_ytd ON sales
FOR INSERT
AS
DECLARE @qty        smallint
DECLARE @title_id   varchar(6)
SELECT @qty = qty,
       @title_id = title_id
FROM   inserted
UPDATE titles
SET    ytd_sales = (ytd_sales + @qty)
WHERE  title_id  = @title_id
GO
```

8. Create a stored procedure called **pr_author_states** which displays the first name, last name, address, and city from the AUTHORS table. The name should be in the format of first name followed by a space followed by the last name. The stored procedure will have **one input parameter** to indicate the state to be selected. If the state is not entered, display a message indicating that a value is required. Use the following code to test your stored procedure to produce the result set listed below.

EXECUTE pr_author_states 'KS'

AuthorID	Name	Address	City
341-22-1782	Meander Smith	10 Mississippi Dr	Lawrence

(1 row(s) affected)

```
CREATE PROCEDURE pr_author_states
( @state char(2) = NULL )
AS
IF @state IS NULL
   BEGIN
     PRINT 'Enter valid state'
   END
ELSE
   BEGIN
     SELECT   au_id                        AS AuthorID,
             (au_fname + ' ' + au_lname) AS Name,
              address                      AS Address,
              city                         AS City
     FROM     authors
     WHERE    state = @state
   END
GO
```

9. Change the **pr_author_states** stored procedure by using the ALTER command to add the state and zip code from the AUTHORS table. Rerun your stored procedure to produce the result set listed below.

EXECUTE pr_author_states 'KS'

AuthorID	Name	Address	City	State	Zip
341-22-1782	Meander Smith	10 Mississippi Dr	Lawrence	KS	66044

(1 row(s) affected)

```
ALTER PROCEDURE pr_author_states
( @state char(2) = NULL )
AS
IF @state IS NULL
   BEGIN
     PRINT 'Enter valid state'
   END
ELSE
   BEGIN
     SELECT   au_id                        AS AuthorID,
             (au_fname + ' ' + au_lname) AS Name,
              address                      AS Address,
              city                         AS City,
              state                        AS State,
              zip                          AS Zip
     FROM     authors
     WHERE    state = @state
   END
GO
```

10. Delete the **pr_author_states** stored procedure.

```
DROP PROCEDURE pr_author_states
```