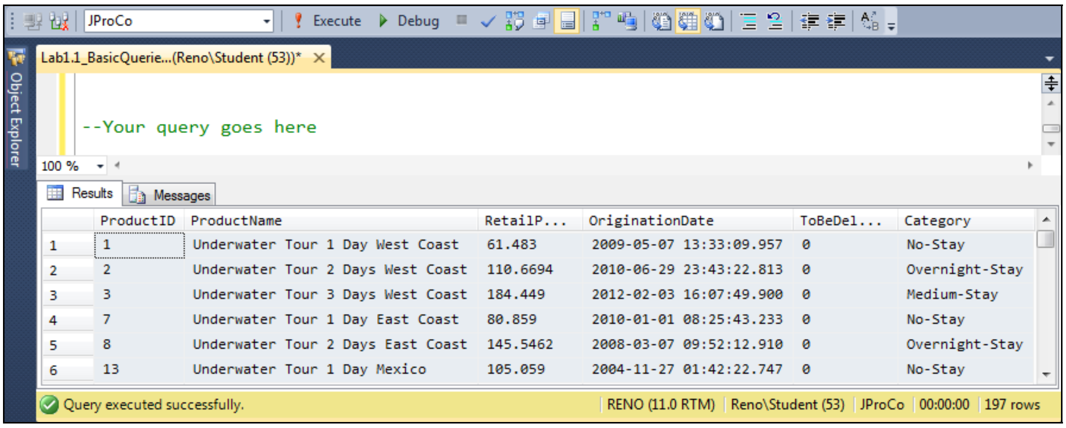
Lab 1.1: Basic Queries

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter1.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Since this is the first lab, please watch the first few videos relating to this book by visiting the www.Joes2Pros.com website.

Skill Check 1: Write a query to show all records from the CurrentProducts table of the JProCo database with a RetailPrice less than $200.00. When done, the result set should resemble Figure 1.19.



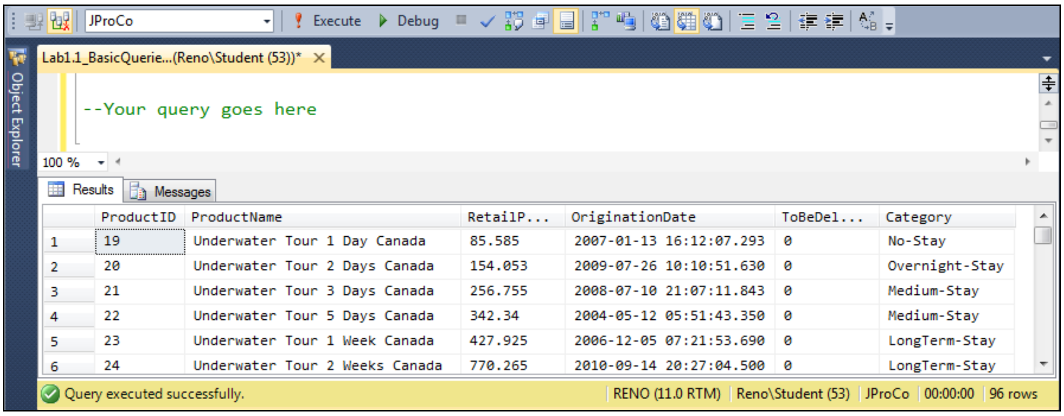
```
--Your query goes here
```

```
   ProductID  ProductName                     RetailP...  OriginationDate          ToBeDel...  Category
1  1          Underwater Tour 1 Day West Coast   61.483    2009-05-07 13:33:09.957  0           No-Stay
2  2          Underwater Tour 2 Days West Coast  110.6694  2010-06-29 23:43:22.813  0           Overnight-Stay
3  3          Underwater Tour 3 Days West Coast  184.449   2012-02-03 16:07:49.900  0           Medium-Stay
4  7          Underwater Tour 1 Day East Coast   80.859    2010-01-01 08:25:43.233  0           No-Stay
5  8          Underwater Tour 2 Days East Coast  145.5462  2008-03-07 09:52:12.910  0           Overnight-Stay
6  13         Underwater Tour 1 Day Mexico       105.059   2004-11-27 01:42:22.747  0           No-Stay
```

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 197 rows
```

Figure 1.19 Skill Check 1 should produce 197 records.

Skill Check 2: Find all the records from the CurrentProducts table that have the word Canada in the ProductName. Show all fields from the CurrentProducts table. When done, the results should resemble Figure 1.20.



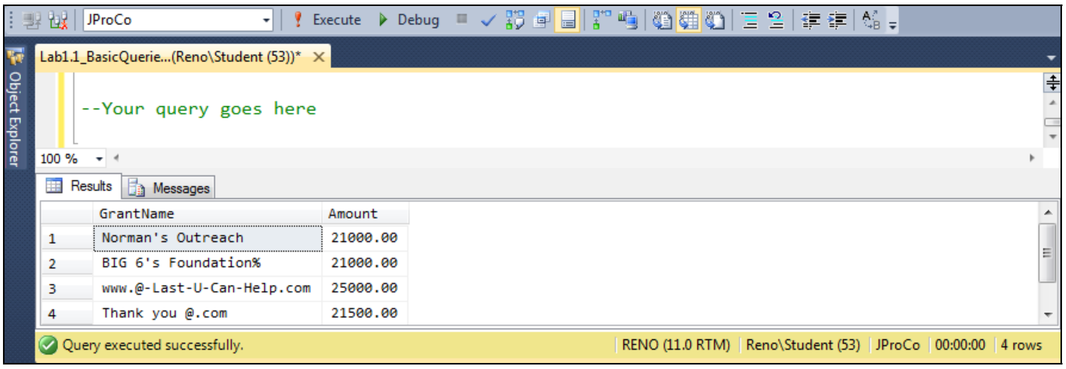
```
--Your query goes here
```

	ProductID	ProductName	RetailP...	OriginationDate	ToBeDel...	Category
1	19	Underwater Tour 1 Day Canada	85.585	2007-01-13 16:12:07.293	0	No-Stay
2	20	Underwater Tour 2 Days Canada	154.053	2009-07-26 10:10:51.630	0	Overnight-Stay
3	21	Underwater Tour 3 Days Canada	256.755	2008-07-10 21:07:11.843	0	Medium-Stay
4	22	Underwater Tour 5 Days Canada	342.34	2004-05-12 05:51:43.350	0	Medium-Stay
5	23	Underwater Tour 1 Week Canada	427.925	2006-12-05 07:21:53.690	0	LongTerm-Stay
6	24	Underwater Tour 2 Weeks Canada	770.265	2010-09-14 20:27:04.500	0	LongTerm-Stay

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 96 rows

Figure 1.20 Skill Check 2 produces 96 records.

Skill Check 3: Grant is a table in the JProCo database. Show all the Grant records with Amount values between 21000 and 30000. Show only the GrantName and Amount fields. When done, the results should resemble Figure 1.21.



```
--Your query goes here
```

	GrantName	Amount
1	Norman's Outreach	21000.00
2	BIG 6's Foundation%	21000.00
3	www.@-Last-U-Can-Help.com	25000.00
4	Thank you @.com	21500.00

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 4 rows

Figure 1.21 Skill Check 3 produces four records.

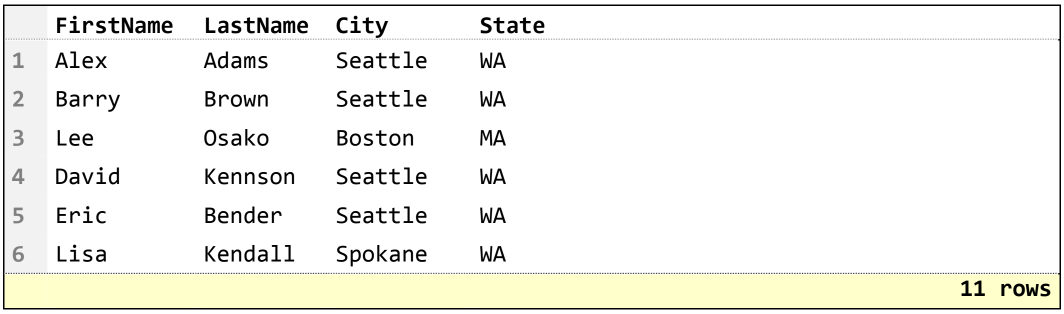
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab1.1_BasicQueries.sql.

Lab 1.2: Joining Tables and Aliases

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter1.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: In a single query, show the employees and cities where they work. Join the Employee and Location tables of the JProCo database on the field they share in common (LocationID). The field selection list should only include FirstName, LastName, City and State. When done, the results should have 11 records and resemble Figure 1.40.

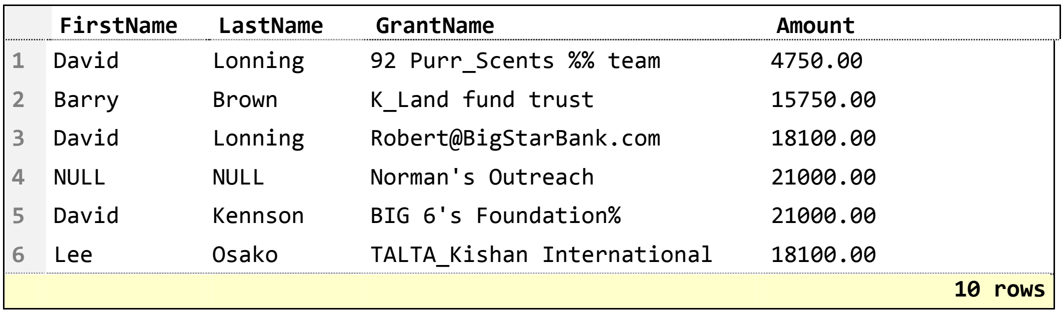


```
   FirstName  LastName  City      State
1  Alex       Adams     Seattle   WA
2  Barry      Brown     Seattle   WA
3  Lee        Osako     Boston    MA
4  David      Kennson   Seattle   WA
5  Eric       Bender    Seattle   WA
6  Lisa       Kendall   Spokane   WA
                                                    11 rows
```

Figure 1.40 Skill Check 1 shows all employees and where they work.

Skill Check 2: Set the database context to JProCo. Write a query that shows a list of records (grants) from the Grant table, plus the first and last names for the employees who acquired them. If an employee has not found a grant, display a NULL where their names would have been.

Accomplish this Skill Check by joining the Employee and Grant tables together. Include the FirstName, LastName, GrantName and Amount fields in the selection list. When done, the results should have 10 records and resemble Figure 1.41.



```
   FirstName  LastName  GrantName                   Amount
1  David      Lonning   92 Purr_Scents %% team      4750.00
2  Barry      Brown     K_Land fund trust           15750.00
3  David      Lonning   Robert@BigStarBank.com      18100.00
4  NULL       NULL      Norman's Outreach           21000.00
5  David      Kennson   BIG 6's Foundation%         21000.00
6  Lee        Osako     TALTA_Kishan International  18100.00
                                                         10 rows
```

Figure 1.41 Skill Check 2 shows all records from Grant table and matching Employee table.

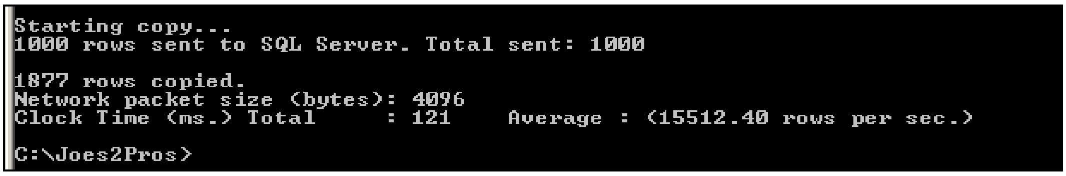
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab1.2_JoiningTables.sql.

Lab 1.3: Using BCP

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter1.3Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Copy the Ch1SalesInvoiceFeed.txt' file to the C:\Joes2Pros folder. Run BCP utility with the correct switches to place these 1877 records into the JProCo.dbo.SalesInvoice table as seen in Figure 1.53.

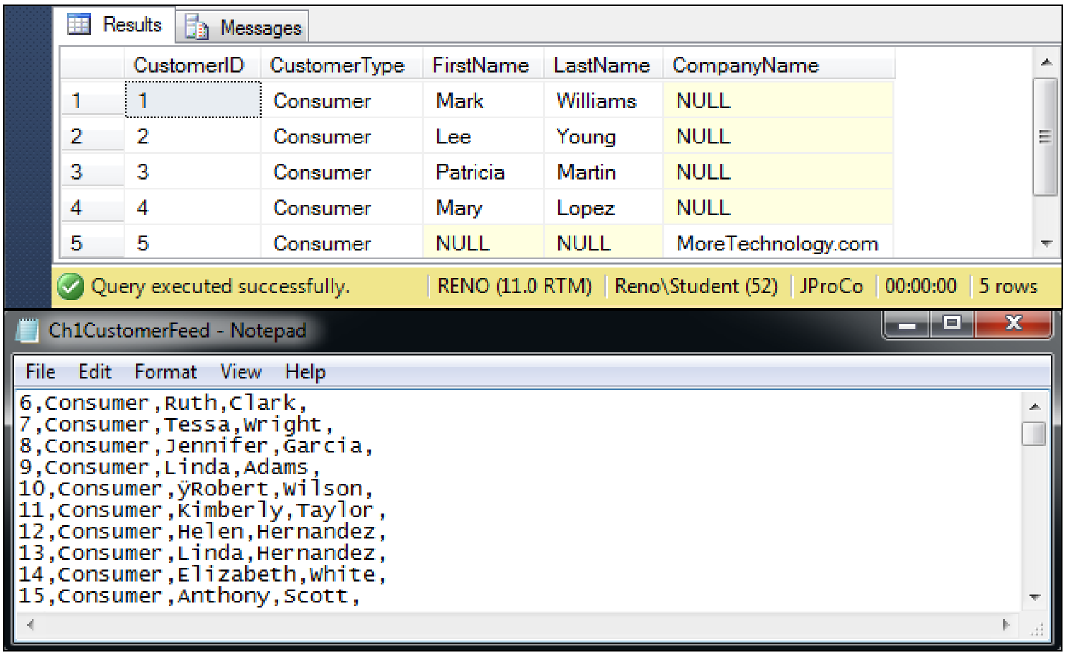


```
Starting copy...
1000 rows sent to SQL Server. Total sent: 1000

1877 rows copied.
Network packet size (bytes): 4096
Clock Time (ms.) Total     : 121     Average : (15512.40 rows per sec.)

C:\Joes2Pros>
```

Figure 1.53 BCP has copied 1877 rows into the SalesInvoice table.

Skill Check 2: The Customer table of JProCo has five test records inside. The Ch1CustomerFeed.txt in the C:\Joes2Pros\Resources folder has 775 verified records.

```
SELECT * FROM Customer
```



Results | Messages

	CustomerID	CustomerType	FirstName	LastName	CompanyName
1	1	Consumer	Mark	Williams	NULL
2	2	Consumer	Lee	Young	NULL
3	3	Consumer	Patricia	Martin	NULL
4	4	Consumer	Mary	Lopez	NULL
5	5	Consumer	NULL	NULL	MoreTechnology.com

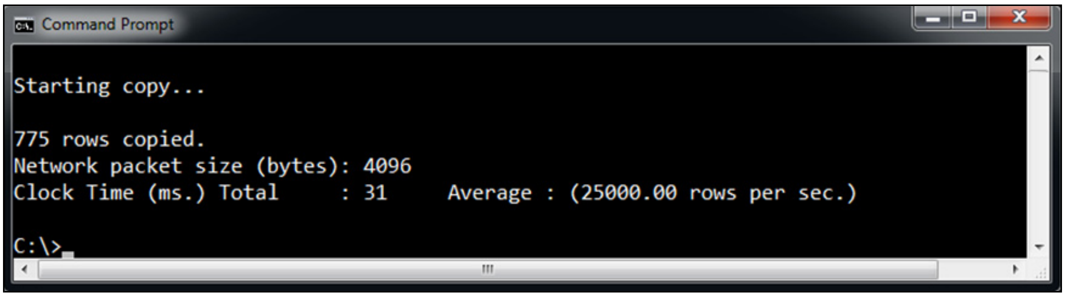
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (52) | JProCo | 00:00:00 | 5 rows

Ch1CustomerFeed - Notepad

```
6,Consumer,Ruth,Clark,
7,Consumer,Tessa,Wright,
8,Consumer,Jennifer,Garcia,
9,Consumer,Linda,Adams,
10,Consumer,ÿRobert,Wilson,
11,Consumer,Kimberly,Taylor,
12,Consumer,Helen,Hernandez,
13,Consumer,Linda,Hernandez,
14,Consumer,Elizabeth,White,
15,Consumer,Anthony,Scott,
```

Figure 1.54 Ch1CustomerFeed.txt is a comma-delimited text file ready to import into the Customer table.

We need to delete the five test records from the Customer table and then import the 775 comma-delimited rows of data from the Ch1CustomerFeed.txt file. When done, the Command Prompt window will resemble the BCP utility results shown here in Figure 1.55.



Command Prompt

```
Starting copy...

775 rows copied.
Network packet size (bytes): 4096
Clock Time (ms.) Total     : 31      Average : (25000.00 rows per sec.)

C:\>_
```

Figure 1.55 BCP shows 775 records have been inserted.

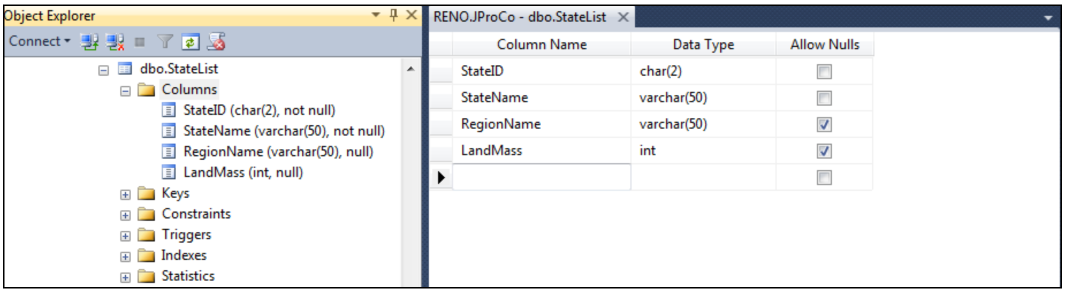
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab1.3_BCP.sql.

Lab 1.4: Creating and Populating Tables

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter1.4Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Create the dbo.StateList table in the JProCo database with the table design shown in Figure 1.59. ***Hint:*** *This table has no primary key.*

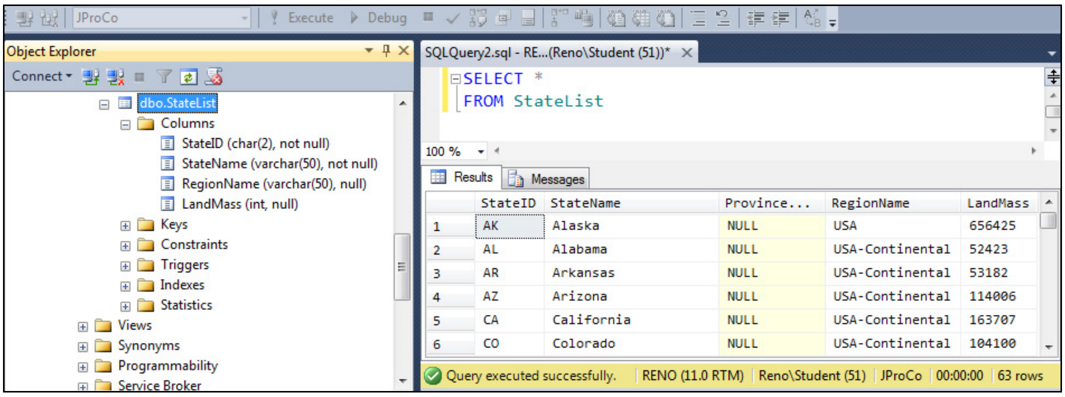


```
dbo.StateList
  Columns
    StateID (char(2), not null)
    StateName (varchar(50), not null)
    RegionName (varchar(50), null)
    LandMass (int, null)
  Keys
  Constraints
  Triggers
  Indexes
  Statistics
```

Column Name	Data Type	Allow Nulls
StateID	char(2)	☐
StateName	varchar(50)	☐
RegionName	varchar(50)	✓
LandMass	int	✓
		☐

Figure 1.59 The design view of the StateList table.

Next use the BCP utility to import Ch1StateListFeed.txt into the StateList table. Verify the results by running a simple query for all fields and records of the StateList table. When done, the results should resemble Figure 1.60.



```
dbo.StateList
  Columns
    StateID (char(2), not null)
    StateName (varchar(50), not null)
    RegionName (varchar(50), null)
    LandMass (int, null)
  Keys
  Constraints
  Triggers
  Indexes
  Statistics
Views
Synonyms
Programmability
Service Broker
```

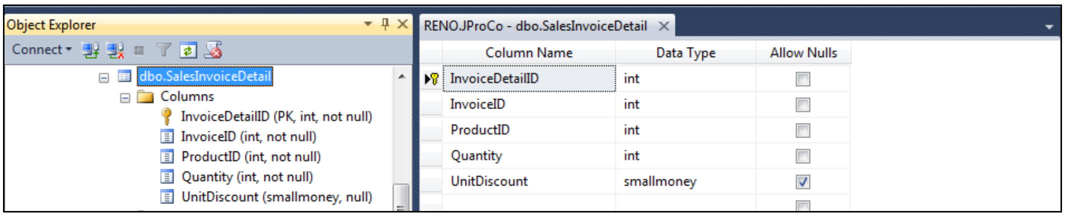
```sql
SELECT *
FROM StateList
```

```
   StateID  StateName   Province...  RegionName        LandMass
1  AK       Alaska      NULL         USA               656425
2  AL       Alabama     NULL         USA-Continental   52423
3  AR       Arkansas    NULL         USA-Continental   53182
4  AZ       Arizona     NULL         USA-Continental   114006
5  CA       California  NULL         USA-Continental   163707
6  CO       Colorado    NULL         USA-Continental   104100
```

```
Query executed successfully.  RENO (11.0 RTM)  Reno\Student (51)  JProCo  00:00:00  63 rows
```

Figure 1.60 The StateList table after importing 53 records from the Ch1StateListFeed.txt file.

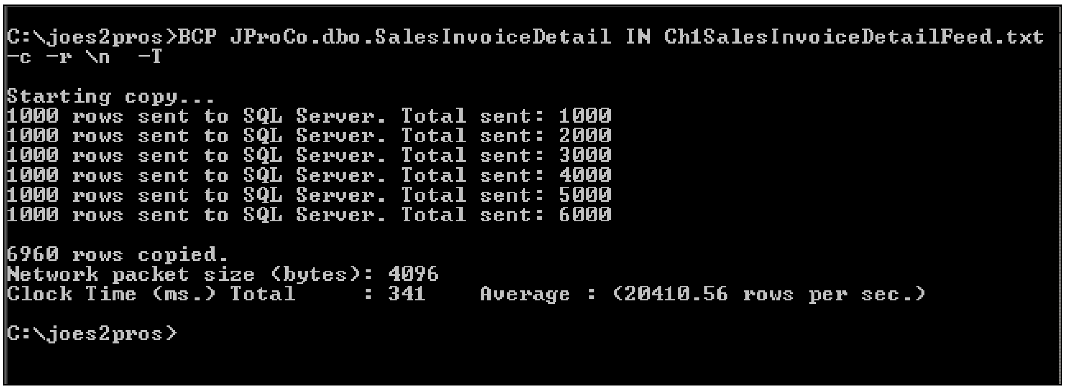
Skill Check 2: Drop, re-create and then populate the SalesInvoiceDetail table in the JProCo database. Use the table design shown in Figure 1.61.


Object Explorer
Connect
dbo.SalesInvoiceDetail
Columns
InvoiceDetailID (PK, int, not null)
InvoiceID (int, not null)
ProductID (int, not null)
Quantity (int, not null)
UnitDiscount (smallmoney, null)
RENO.JProCo - dbo.SalesInvoiceDetail

Column Name	Data Type	Allow Nulls
InvoiceDetailID	int	☐
InvoiceID	int	☐
ProductID	int	☐
Quantity	int	☐
UnitDiscount	smallmoney	✓

Figure 1.61 The design view of SalesInvoiceDetail.

Populate the table with the Ch1SalesInvoiceDetailFeed.txt using the BCP utility. During the import into the database, the results will look like Figure 1.62.

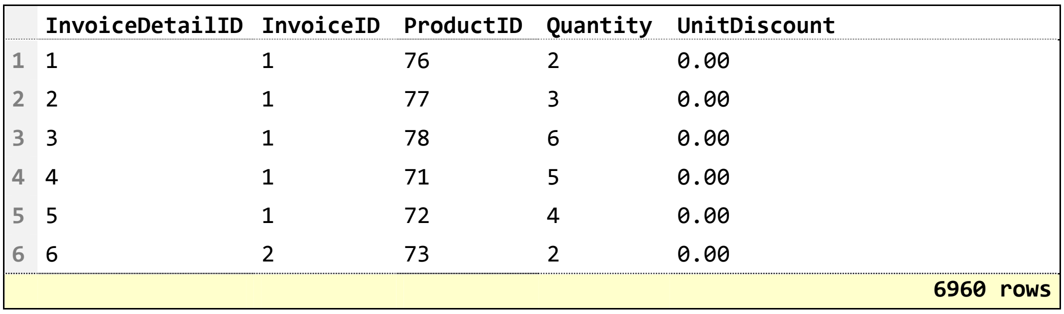


```
C:\joes2pros>BCP JProCo.dbo.SalesInvoiceDetail IN Ch1SalesInvoiceDetailFeed.txt
-c -r \n  -T

Starting copy...
1000 rows sent to SQL Server. Total sent: 1000
1000 rows sent to SQL Server. Total sent: 2000
1000 rows sent to SQL Server. Total sent: 3000
1000 rows sent to SQL Server. Total sent: 4000
1000 rows sent to SQL Server. Total sent: 5000
1000 rows sent to SQL Server. Total sent: 6000

6960 rows copied.
Network packet size (bytes): 4096
Clock Time (ms.) Total     : 341     Average : (20410.56 rows per sec.)

C:\joes2pros>
```

Figure 1.62 BCP copied 6960 records into the SalesInvoiceDetail table.

A quick check of the table after importing the data should verify the results shown in Figure 1.63.

```
SELECT *
FROM SalesInvoiceDetail
```



	InvoiceDetailID	InvoiceID	ProductID	Quantity	UnitDiscount
1	1	1	76	2	0.00
2	2	1	77	3	0.00
3	3	1	78	6	0.00
4	4	1	71	5	0.00
5	5	1	72	4	0.00
6	6	2	73	2	0.00

6960 rows

Figure 1.63 A simple query of SalesInvoiceDetail shows the table is now populated.

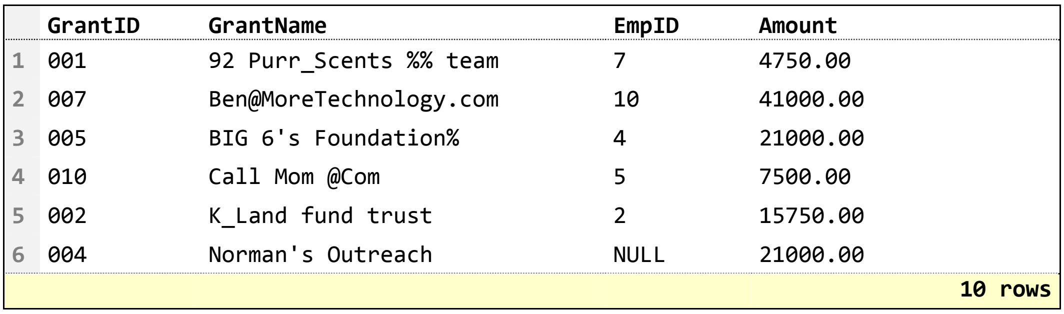
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab1.4_CreatingAndPopulatingTables.sql

Lab 2.1: Sorting Data

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter2.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

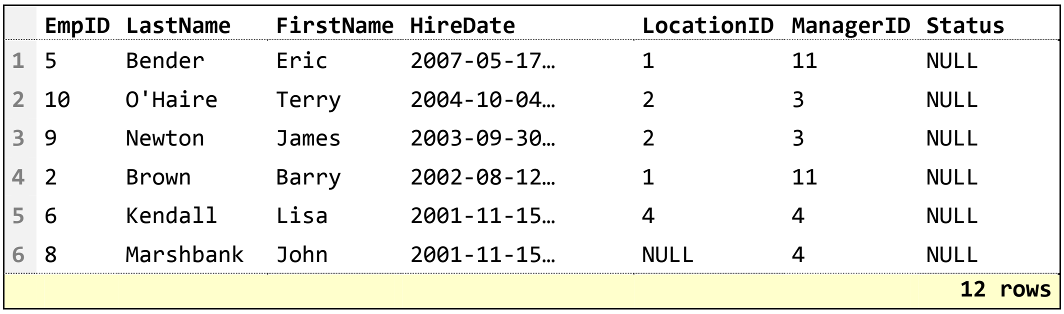
Skill Check 1: Show all records from the Grant table sorted alphabetically by GrantName. Your result should look like Figure 2.9.



```
  GrantID   GrantName                  EmpID   Amount
1 001       92 Purr_Scents %% team     7       4750.00
2 007       Ben@MoreTechnology.com     10      41000.00
3 005       BIG 6's Foundation%        4       21000.00
4 010       Call Mom @Com              5       7500.00
5 002       K_Land fund trust          2       15750.00
6 004       Norman's Outreach          NULL    21000.00
                                                          10 rows
```

Figure 2.9 The result of Skill Check 1 shows 10 records sorted by GrantName.

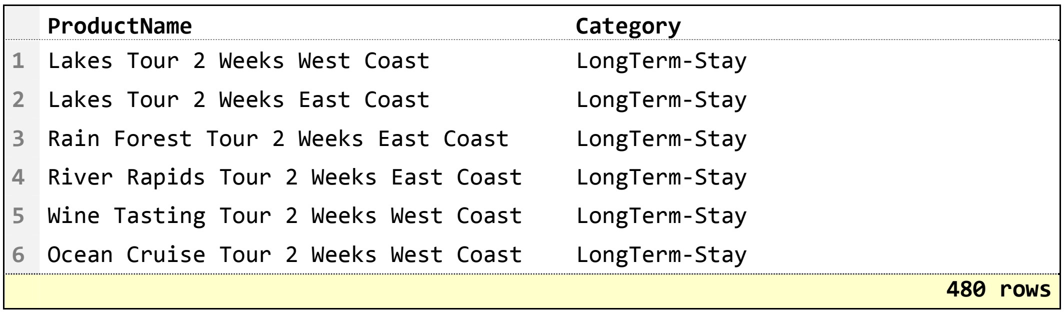
Skill Check 2: Show all fields from the Employee table. The most recent HireDate should appear first (Figure 2.10).



```
  EmpID LastName  FirstName HireDate      LocationID ManagerID Status
1 5     Bender    Eric      2007-05-17…   1          11        NULL
2 10    O'Haire   Terry     2004-10-04…   2          3         NULL
3 9     Newton    James     2003-09-30…   2          3         NULL
4 2     Brown     Barry     2002-08-12…   1          11        NULL
5 6     Kendall   Lisa      2001-11-15…   4          4         NULL
6 8     Marshbank John      2001-11-15…   NULL       4         NULL
                                                             12 rows
```

Figure 2.10 Skill Check 2 shows the most recently hired person listed first.

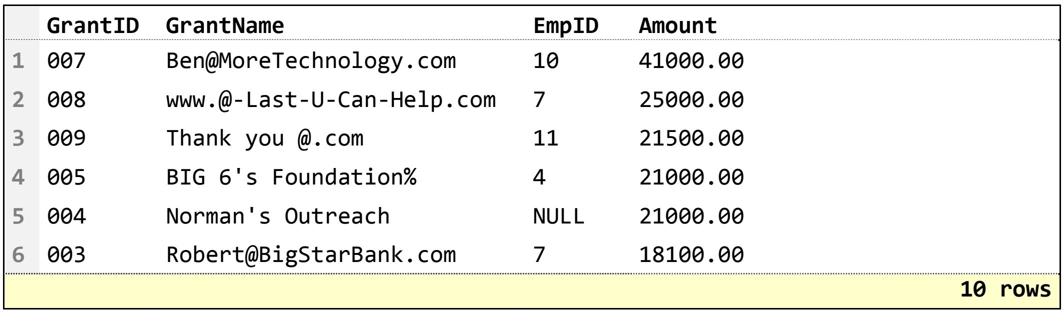
Skill Check 3: Query the CurrentProducts table for just the ProductName and Category fields. Sort the table by the most expensive RetailPrice on top and the least on the bottom. When you're done, your result should resemble the figure you see here (Figure 2.11 shows the first 6 of 480 rows):



```
  ProductName                          Category
1 Lakes Tour 2 Weeks West Coast        LongTerm-Stay
2 Lakes Tour 2 Weeks East Coast        LongTerm-Stay
3 Rain Forest Tour 2 Weeks East Coast  LongTerm-Stay
4 River Rapids Tour 2 Weeks East Coast LongTerm-Stay
5 Wine Tasting Tour 2 Weeks West Coast LongTerm-Stay
6 Ocean Cruise Tour 2 Weeks West Coast LongTerm-Stay
                                                  480 rows
```

Figure 2.11 Shows the most expensive products first without showing RetailPrice in the SELECT list.

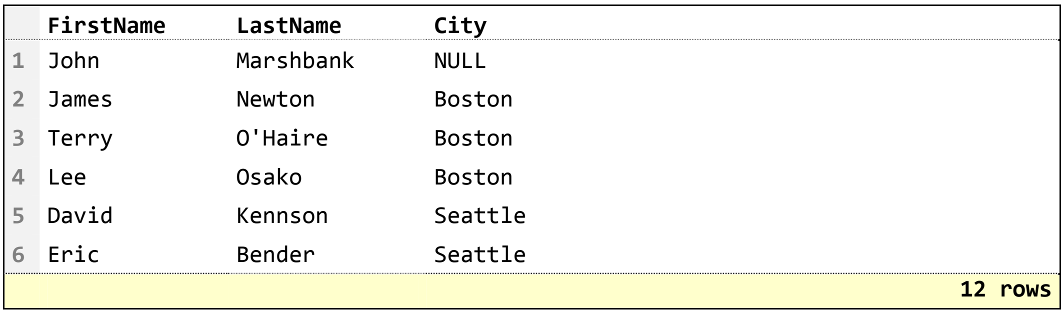
Skill Check 4: Now sort all the fields of the Grant table from highest to lowest amount. If any Grants have a tying amount, then list the ties alphabetically by GrantName (Figure 2.12). Amount ($21,000), Big 6 is listed before Norman's because the secondary sort is alphabetical by GrantName.



```
   GrantID  GrantName                  EmpID  Amount
1  007      Ben@MoreTechnology.com     10     41000.00
2  008      www.@-Last-U-Can-Help.com  7      25000.00
3  009      Thank you @.com            11     21500.00
4  005      BIG 6's Foundation%        4      21000.00
5  004      Norman's Outreach          NULL   21000.00
6  003      Robert@BigStarBank.com     7      18100.00
                                                                    10 rows
```

Figure 2.12 Skill Check 4 shows the highest amounts listed first. Where two grants have the same value the first GrantName value shows up first in an A to Z sort.

Skill Check 5: Join the Employee and Location tables together in an OUTER JOIN that shows all the employee records even if they have no location. Show the fields FirstName, LastName and City. Sort your result so that NULL City names appear first and the remaining values appear in ascending order (Figure 2.13).



```
   FirstName  LastName   City
1  John       Marshbank  NULL
2  James      Newton     Boston
3  Terry      O'Haire    Boston
4  Lee        Osako      Boston
5  David      Kennson    Seattle
6  Eric       Bender     Seattle
                                                                    12 rows
```

Figure 2.13 Skill Check 5 shows Employee names and City listed in order by City with NULLs appearing first.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab2.1_SortingData.sql

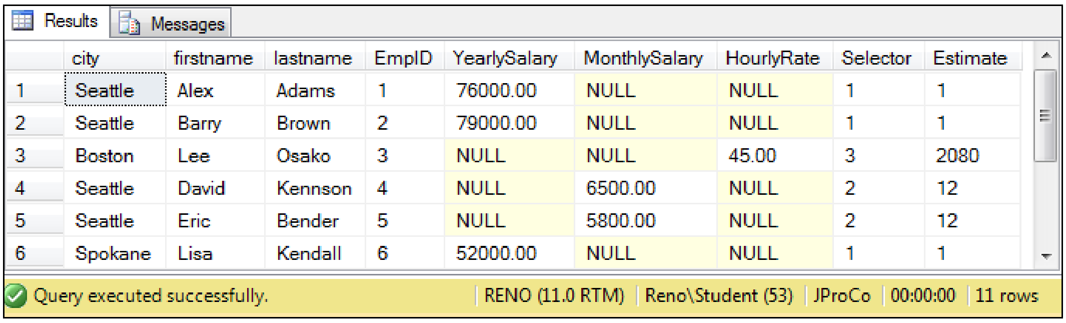
Lab 2.2: Three Table Query

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter2.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Show all the city names and rates of pay for each employee in those cities. You will need to join the Location, Employee and PayRates tables. Show the City field from the Location table. Include FirstName and LastName from the Employee table and all fields from the PayRates table. When you're done, your result should resemble Figure 2.20.

```
SELECT lo.City, em.FirstName, em.LastName, pr.*
--Remaining Code Here
```



	city	firstname	lastname	EmpID	YearlySalary	MonthlySalary	HourlyRate	Selector	Estimate
1	Seattle	Alex	Adams	1	76000.00	NULL	NULL	1	1
2	Seattle	Barry	Brown	2	79000.00	NULL	NULL	1	1
3	Boston	Lee	Osako	3	NULL	NULL	45.00	3	2080
4	Seattle	David	Kennson	4	NULL	6500.00	NULL	2	12
5	Seattle	Eric	Bender	5	NULL	5800.00	NULL	2	12
6	Spokane	Lisa	Kendall	6	52000.00	NULL	NULL	1	1

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 11 rows
```

Figure 2.20 A query joining the Location, Employee and PayRates tables together.

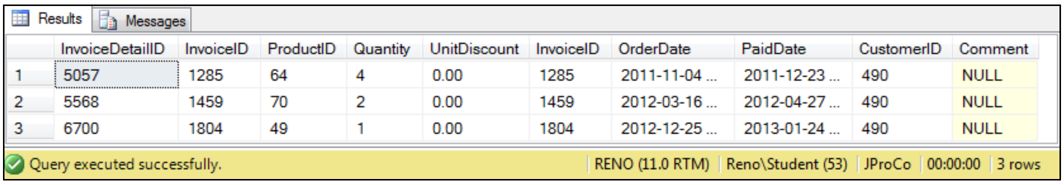
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab2.2_ThreeTableQuery.sql

Lab 2.3: Many-to-Many Relationships

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter2.3Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: In JProCo write a query that shows all the invoices ordered by customer 490. Show all fields from just 2 tables (SalesInvoice and SalesInvoiceDetail). When you're done your result should resemble Figure 2.29.



	InvoiceDetailID	InvoiceID	ProductID	Quantity	UnitDiscount	InvoiceID	OrderDate	PaidDate	CustomerID	Comment
1	5057	1285	64	4	0.00	1285	2011-11-04 ...	2011-12-23 ...	490	NULL
2	5568	1459	70	2	0.00	1459	2012-03-16 ...	2012-04-27 ...	490	NULL
3	6700	1804	49	1	0.00	1804	2012-12-25 ...	2013-01-24 ...	490	NULL

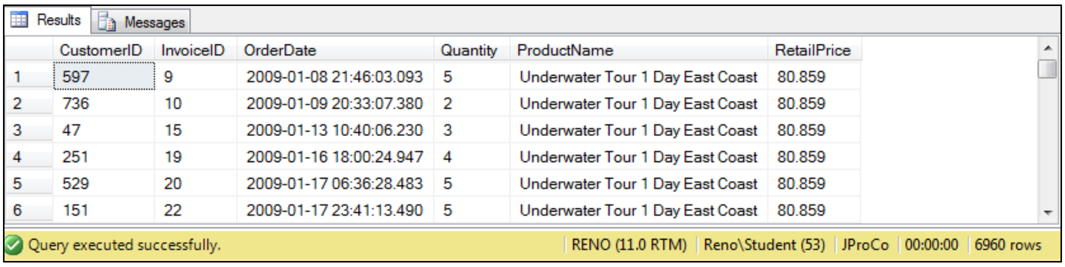
```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 3 rows
```

Figure 2.29 Skill Check 1 shows the Customer who ordered each invoice.

Skill Check 2: Write a query that combines SalesInvoice, SalesInvoiceDetail and CurrentProducts. Show the following fields:

- SalesInvoice.CustomerID
- SalesInvoice.InvoiceID
- SalesInvoice.OrderDate
- SalesInvoiceDetail.Quantity
- CurrentProducts.ProductName
- CurrentProducts.RetailPrice

When you are done your result should resemble Figure 2.30.



	CustomerID	InvoiceID	OrderDate	Quantity	ProductName	RetailPrice
1	597	9	2009-01-08 21:46:03.093	5	Underwater Tour 1 Day East Coast	80.859
2	736	10	2009-01-09 20:33:07.380	2	Underwater Tour 1 Day East Coast	80.859
3	47	15	2009-01-13 10:40:06.230	3	Underwater Tour 1 Day East Coast	80.859
4	251	19	2009-01-16 18:00:24.947	4	Underwater Tour 1 Day East Coast	80.859
5	529	20	2009-01-17 06:36:28.483	5	Underwater Tour 1 Day East Coast	80.859
6	151	22	2009-01-17 23:41:13.490	5	Underwater Tour 1 Day East Coast	80.859

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 6960 rows
```

Figure 2.30 Skill Check 2 shows all the product details for each InvoiceID.

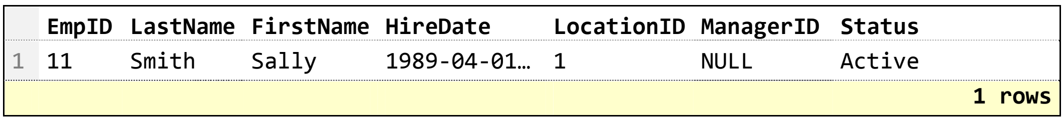
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab2.3_Many-to-Many_Relationships.sql

Lab 3.1: Working With NULLs

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter3.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: The Employee table in JProCo has a field called ManagerID. Write a query to show all Employees who don't have a ManagerID. When you're done, your result should resemble Figure 3.3.

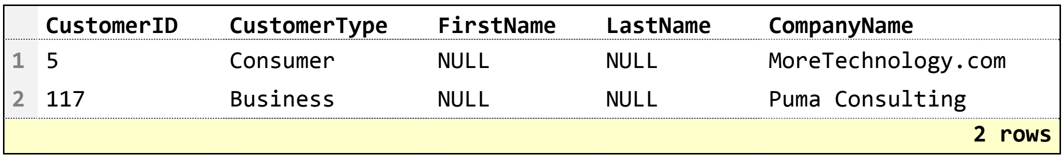


	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	11	Smith	Sally	1989-04-01…	1	NULL	Active

1 rows

Figure 3.3 Skill Check 1 shows all JProCo Employees who have no ManagerID.

Skill Check 2: Find the two records in the Customer table where the company name is not NULL, as shown in Figure 3.4.

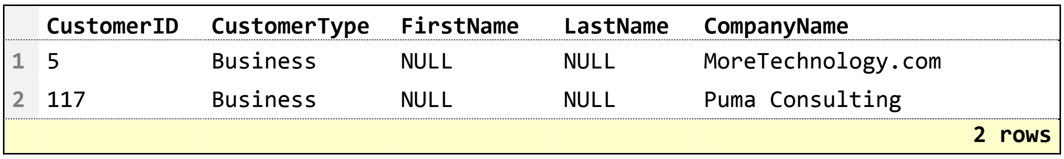


	CustomerID	CustomerType	FirstName	LastName	CompanyName
1	5	Consumer	NULL	NULL	MoreTechnology.com
2	117	Business	NULL	NULL	Puma Consulting

2 rows

Figure 3.4 Find the 2 records in the Customer table where the field CompanyName is not NULL.

Skill Check 3: Using your result set from Skill Check 2, write a statement to update the CustomerType field of the Customer table to "Business" for each record where the company name is not NULL. When you are done, you should have two records with the value "Business" in the CustomerType field (Figure 3.5).

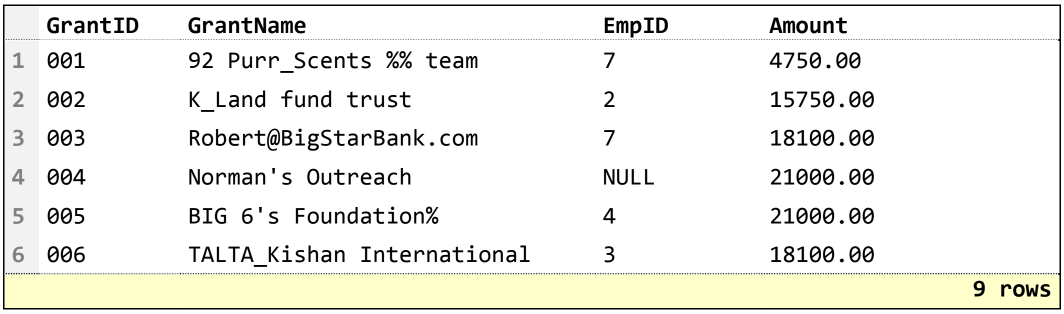


	CustomerID	CustomerType	FirstName	LastName	CompanyName
1	5	Business	NULL	NULL	MoreTechnology.com
2	117	Business	NULL	NULL	Puma Consulting

2 rows

Figure 3.5 Skill Check 3 changes the CustomerType to "Business" if CompanyName is not NULL.

Skill Check 4: You have 10 grants in your Grant table. One grant was procured by EmpID 5 and the other nine grants were not. Show all grants found by employees other than EmpID 5, as well as all grants for which EmpID is NULL. When you're done, your result should resemble Figure 3.6.



```
  GrantID   GrantName                     EmpID      Amount
1 001       92 Purr_Scents %% team        7          4750.00
2 002       K_Land fund trust             2          15750.00
3 003       Robert@BigStarBank.com        7          18100.00
4 004       Norman's Outreach             NULL       21000.00
5 005       BIG 6's Foundation%           4          21000.00
6 006       TALTA_Kishan International    3          18100.00
                                                               9 rows
```

Figure 3.6 Skill Check 4 shows the nine grants not found by EmpID 5.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab3.1_WorkingWithNulls.sql

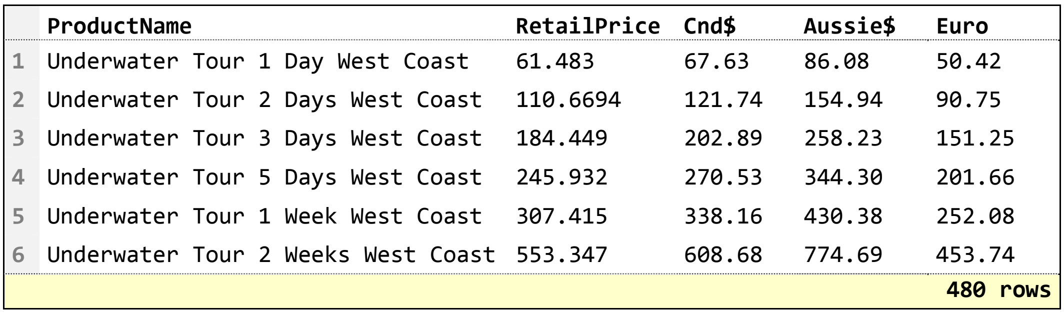
Lab 3.2: Expression Fields

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter3.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Using the CurrentProducts table of JProCo, make another field to express the price in Canadian currency called CDN$ that is 1.1 times the stated RetailPrice in JProCo's CurrentProducts table. Then create two additional fields named Aussie$ (1.4 times the RetailPrice) and Euro (which is .82 times the RetailPrice). When you're done your result should resemble Figure 3.16.

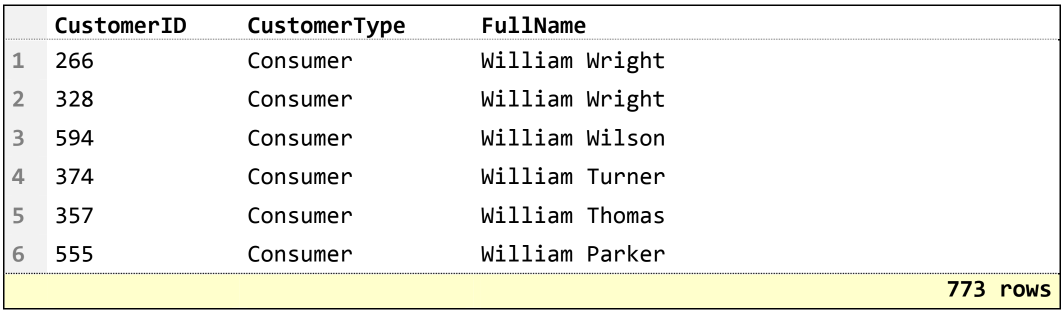
```sql
SELECT ProductName, RetailPrice,
--Remaining Code Here
```



```
   ProductName                     RetailPrice  Cnd$     Aussie$  Euro
1  Underwater Tour 1 Day West Coast    61.483       67.63    86.08    50.42
2  Underwater Tour 2 Days West Coast   110.6694     121.74   154.94   90.75
3  Underwater Tour 3 Days West Coast   184.449      202.89   258.23   151.25
4  Underwater Tour 5 Days West Coast   245.932      270.53   344.30   201.66
5  Underwater Tour 1 Week West Coast   307.415      338.16   430.38   252.08
6  Underwater Tour 2 Weeks West Coast  553.347      608.68   774.69   453.74
                                                                  480 rows
```

Figure 3.16 Three new currency columns calculated from RetailPrice (US dollars).

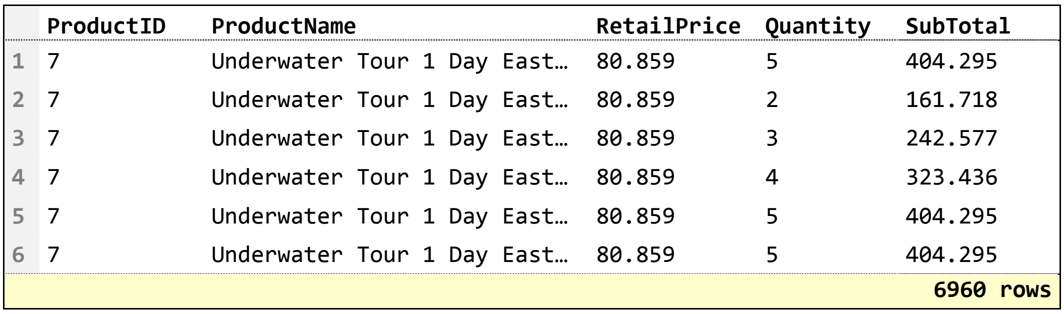
Skill Check 2: In JProCo find the 773 records in your Customer table where the CustomerType is Consumer. Show the CustomerID, CustomerType field and the FullName expression field. Your results should be sorted by the FullName field (Z-A). When you're done your result should resemble Figure 3.17.



```
   CustomerID   CustomerType   FullName
1  266          Consumer       William Wright
2  328          Consumer       William Wright
3  594          Consumer       William Wilson
4  374          Consumer       William Turner
5  357          Consumer       William Thomas
6  555          Consumer       William Parker
                                                        773 rows
```

Figure 3.17 Add the expression field FullName.

Skill Check 3: Join the SalesInvoiceDetail table to the CurrentProducts table. Show the ProductID, ProductName and RetailPrice from the CurrentProducts table. Show Quantity from the SalesInvoiceDetail table. Create an expression field called SubTotal which multiplies RetailPrice by Quantity. Your result should look like Figure 3.18.

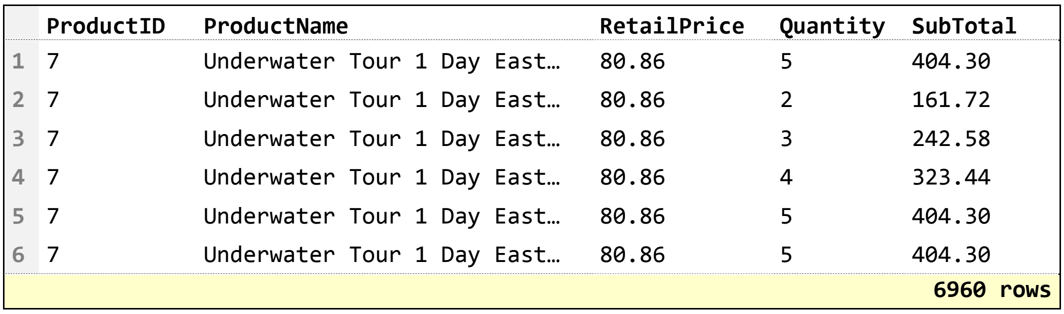


```
   ProductID  ProductName                RetailPrice  Quantity  SubTotal
1  7          Underwater Tour 1 Day East…  80.859     5         404.295
2  7          Underwater Tour 1 Day East…  80.859     2         161.718
3  7          Underwater Tour 1 Day East…  80.859     3         242.577
4  7          Underwater Tour 1 Day East…  80.859     4         323.436
5  7          Underwater Tour 1 Day East…  80.859     5         404.295
6  7          Underwater Tour 1 Day East…  80.859     5         404.295
                                                              6960 rows
```

Figure 3.18 Skill Check 3 adds dynamic field SubTotal.

Skill Check 4: Modify your query from Skill Check 3. Using the Round function, show Retail Price and the SubTotal expression field rounded to the nearest penny. When you're done, your result will resemble Figure 3.19.

```
SELECT cp.ProductID, cp.ProductName,
--Remaining Code Here
```



```
   ProductID   ProductName                    RetailPrice   Quantity   SubTotal
1  7           Underwater Tour 1 Day East…    80.86         5          404.30
2  7           Underwater Tour 1 Day East…    80.86         2          161.72
3  7           Underwater Tour 1 Day East…    80.86         3          242.58
4  7           Underwater Tour 1 Day East…    80.86         4          323.44
5  7           Underwater Tour 1 Day East…    80.86         5          404.30
6  7           Underwater Tour 1 Day East…    80.86         5          404.30
                                                                      6960 rows
```

Figure 3.19 Use the ROUND function to display Skill Check 3 results rounded to the nearest penny.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab3.2_ExpressionFields.sql.

Lab 3.3: Identity Fields

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

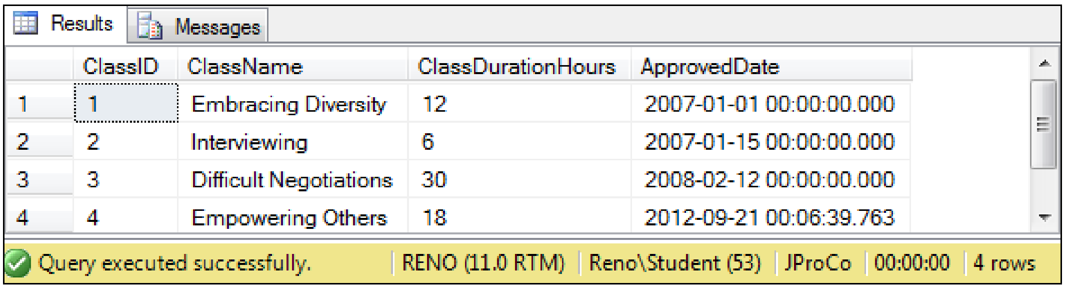
Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter3.3Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: The MgmtTraining table in JProCo has ClassID as an identity field. Previously, this table had many records deleted from it. We want to insert a value of 'Empowering Others' in the ClassName field with a ClassID of 4.

If we run a simple INSERT statement, the identity counter is already past the number 4. We must set the table's property for inserting values to allow manually inserting all fields for this record. The ApprovedDate field should be set using the CURRENT_TIMESTAMP property.

When done, the results should resemble those shown in Figure 3.34.

```
--Skill Check 1 Code Here
SELECT * FROM MgmtTraining
```



	ClassID	ClassName	ClassDurationHours	ApprovedDate
1	1	Embracing Diversity	12	2007-01-01 00:00:00.000
2	2	Interviewing	6	2007-01-15 00:00:00.000
3	3	Difficult Negotiations	30	2008-02-12 00:00:00.000
4	4	Empowering Others	18	2012-09-21 00:06:39.763

```
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 4 rows
```

Figure 3.34 Manually inserting ClassID 4 "Empowering Others" into the MgmtTraining table.

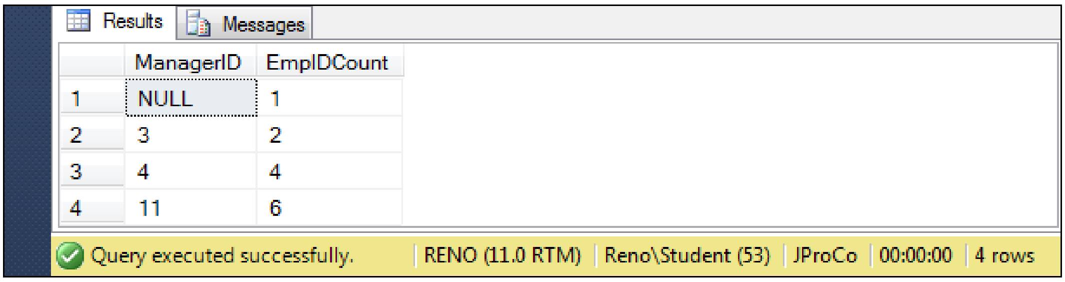
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab3.3_IdentityFields.sql.

Lab 4.1: Using GROUP BY

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter4.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Query the Employee table of JProCo to see how many people work for each ManagerID. Select the ManagerID and Count the EmpID field. Alias the field as EmpIDCount. When you're done, your result should resemble the figure you see here (Figure 4.18).

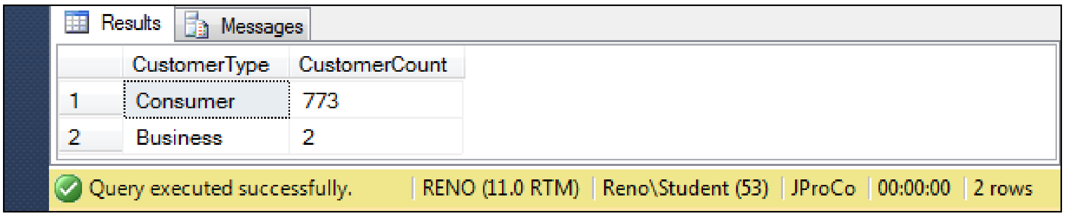


	ManagerID	EmpIDCount
1	NULL	1
2	3	2
3	4	4
4	11	6

```
Query executed successfully.  | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 4 rows
```

Figure 4.18 Skill Check 1 displays the count of people associated with each ManagerID.

Skill Check 2: Perform a grouping query on the Customer table to get a count of how many consumers versus Business customers you have. Alias the field as CustomerCount. Group on the CustomerType field. When you're done your result should resemble the figure you see here (Figure 4.19).

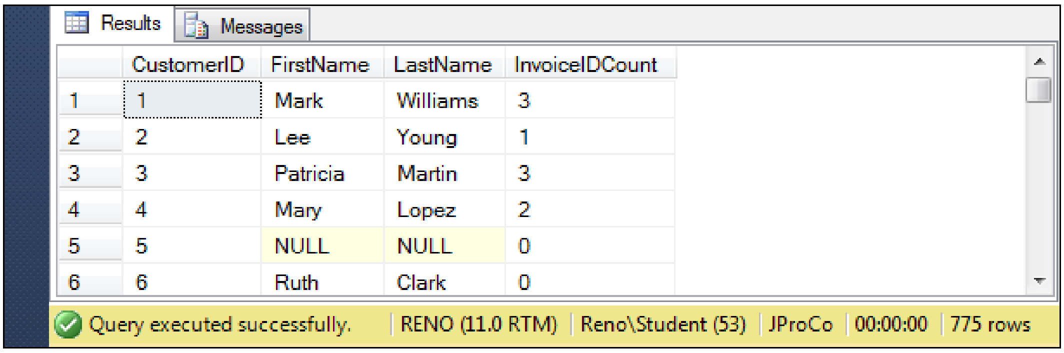


	CustomerType	CustomerCount
1	Consumer	773
2	Business	2

```
Query executed successfully.  | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 2 rows
```

Figure 4.19 Group on CustomerType to show numbers of Consumer vs. Business customers.

Skill Check 3: Get a list of all Customers and how many Invoice orders each one has placed. You will need to join the Customer and SalesInvoice tables. Alias the aggregated field as InvoiceIDCount. If a Customer has not ordered yet, then you should still see their name with a zero next to it. Hint: this will require an OUTER JOIN between Customer and SalesInvoice (Figure 4.20). Your results should have 775 records.

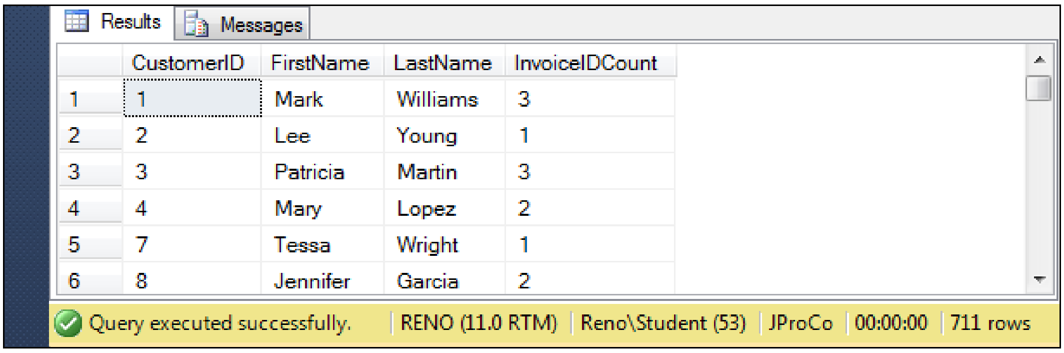


	CustomerID	FirstName	LastName	InvoiceIDCount
1	1	Mark	Williams	3
2	2	Lee	Young	1
3	3	Patricia	Martin	3
4	4	Mary	Lopez	2
5	5	NULL	NULL	0
6	6	Ruth	Clark	0

```
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 775 rows
```

Figure 4.20 Skill Check 3 shows each customer and the COUNT of their invoices.

Skill Check 4: Make a slight modification to Skill Check 3 so that only Customers who have placed at least one order appear in the query (Hint: change the type of join). Notice CustomerID 5 does not appear in this result.



	CustomerID	FirstName	LastName	InvoiceIDCount
1	1	Mark	Williams	3
2	2	Lee	Young	1
3	3	Patricia	Martin	3
4	4	Mary	Lopez	2
5	7	Tessa	Wright	1
6	8	Jennifer	Garcia	2

```
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 711 rows
```

Figure 4.21 Skill Check 4 shows only customers who placed at least one order.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab4.1_UsingGroupBy.sql.

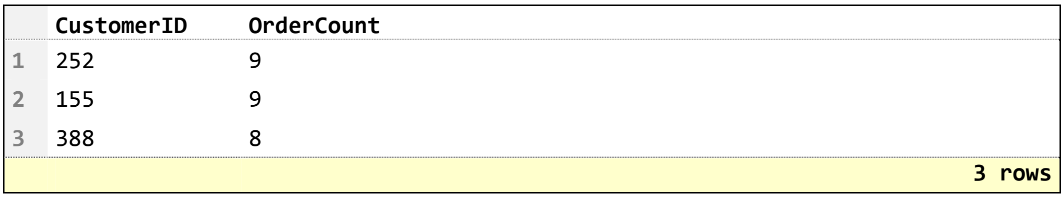
Lab 4.2: Filtering Aggregated Results

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter4.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Using the SalesInvoice table, write a query which groups on CustomerID and counts the number of orders (also called invoices) each customer has made. Return only the records where the CustomerID has ordered more than 7 times. Show the aggregated field as OrderCount.

When done, the results should resemble Figure 4.37.

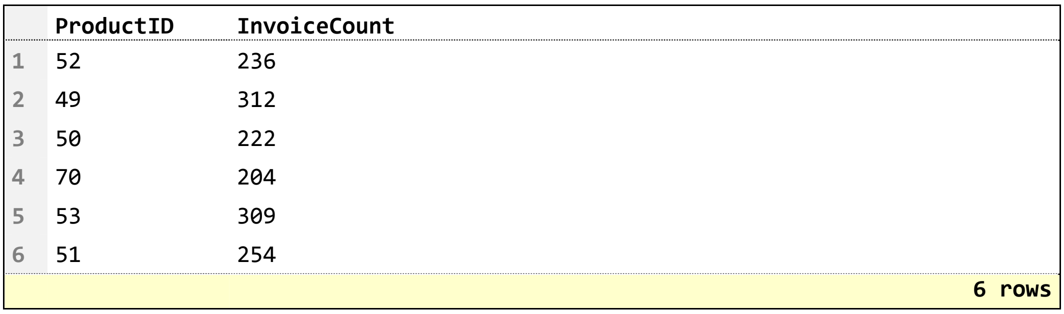


```
   CustomerID    OrderCount
1  252           9
2  155           9
3  388           8
                                  3 rows
```

Figure 4.37 Skill Check 1 results show CustomerIDs with more than 7 orders.

Skill Check 2: Query the SalesInvoiceDetail table and show just the ProductID and InvoiceID fields. Change the query to group on ProductID and count the InvoiceID field. Return only the records where the ProductID has been ordered more than 200 times. When done the result should resemble Figure 4.38.

There is no need to join to the CurrentProducts table since we will be grouping on ProductID which is already in the SalesInvoiceDetail table.



```
   ProductID    InvoiceCount
1  52           236
2  49           312
3  50           222
4  70           204
5  53           309
6  51           254
                                  6 rows
```

Figure 4.38 Skill Check 2 looks for products ordered more than 200 times.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab4.2_FilteringAggregatedResults.sql.

Lab 4.3: Aggregation in Stored Procedures

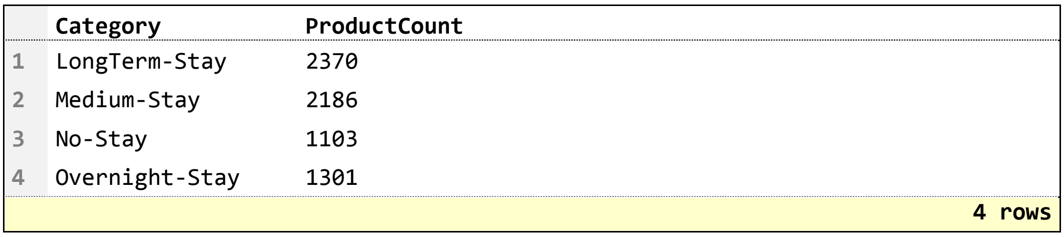
Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter4.3Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Using what you've learned, create a stored procedure called GetCategoriesByProductCount. This will join the CurrentProducts and the SalesInvoiceDetail tables and show the total number of orders for each Category.

When done, execute the stored procedure and view the results. The results should resemble those shown in Figure 4.44.

```sql
EXEC GetCategoriesByProductCount
```



```
   Category          ProductCount
1  LongTerm-Stay     2370
2  Medium-Stay       2186
3  No-Stay           1103
4  Overnight-Stay    1301
                                          4 rows
```

Figure 4.44 Skill Check 1 creates the stored procedure GetCategoriesByProductCount.

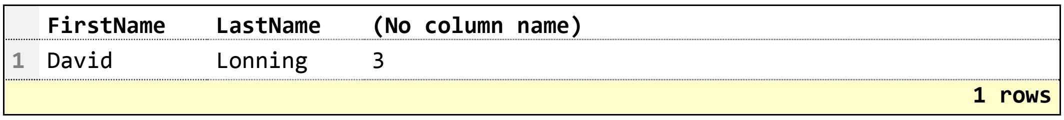
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab4.3_AggregationInStoredProcedures.sql.

Lab 5.1: Finding Duplicates

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter5.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

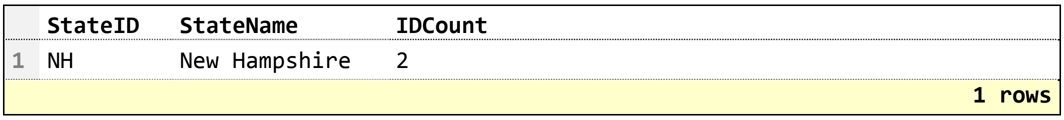
Skill Check 1: Go to the JProCo database and count all employees having multiple grants listed in the Grant table. Your result should resemble Figure 5.7.



```
   FirstName   LastName   (No column name)
1  David       Lonning    3
                                                    1 rows
```

Figure 5.7 Skill Check 1 looks for employees with multiple entries in the Grant table.

Skill Check 2: Query the StateList table to find any duplicate records. List all duplicated StateID values you find. Title your aggregated field IDCount. When you're done, your result should resemble Figure 5.8.



```
   StateID   StateName       IDCount
1  NH        New Hampshire   2
                                                    1 rows
```

Figure 5.8 Find records with a duplicate StateID in the table JProCo.dbo.StateList.

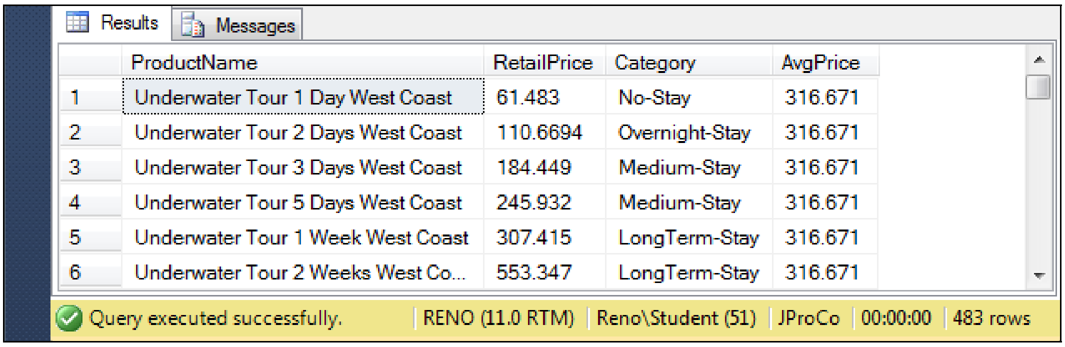
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab5.1_FindingDuplicates.sql.

Lab 5.2: The OVER Clause

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter5.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Use the JProCo database and query the CurrentProducts table for the fields ProductName, RetailPrice and Category. Create an expression field that combines AVG() with an OVER() clause and call it AvgPrice. When you are done, your result should resemble the figure you see here (Figure 5.26).

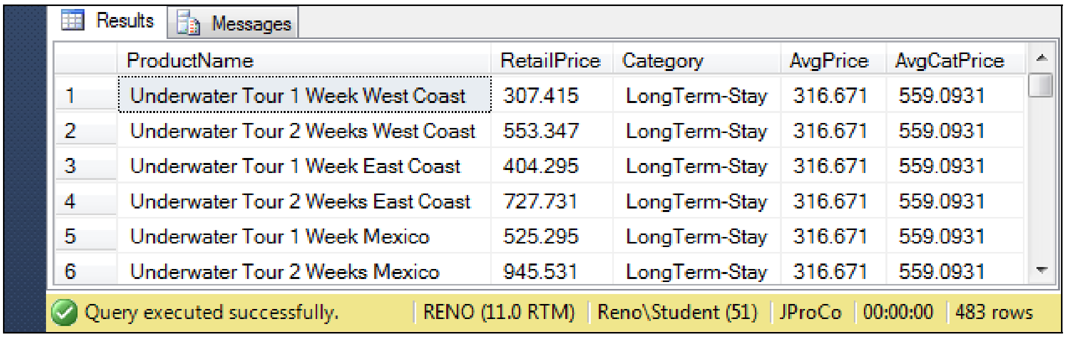


	ProductName	RetailPrice	Category	AvgPrice
1	Underwater Tour 1 Day West Coast	61.483	No-Stay	316.671
2	Underwater Tour 2 Days West Coast	110.6694	Overnight-Stay	316.671
3	Underwater Tour 3 Days West Coast	184.449	Medium-Stay	316.671
4	Underwater Tour 5 Days West Coast	245.932	Medium-Stay	316.671
5	Underwater Tour 1 Week West Coast	307.415	LongTerm-Stay	316.671
6	Underwater Tour 2 Weeks West Co...	553.347	LongTerm-Stay	316.671

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 483 rows

Figure 5.26 Skill Check 1 creates a new expression field combining AVG() with an OVER clause.

Skill Check 2: Take the query from Skill Check 1 and add another expression field called AvgCatPrice that shows the average price for the Category for any given product. When you are done, your result will resemble Figure 5.27.

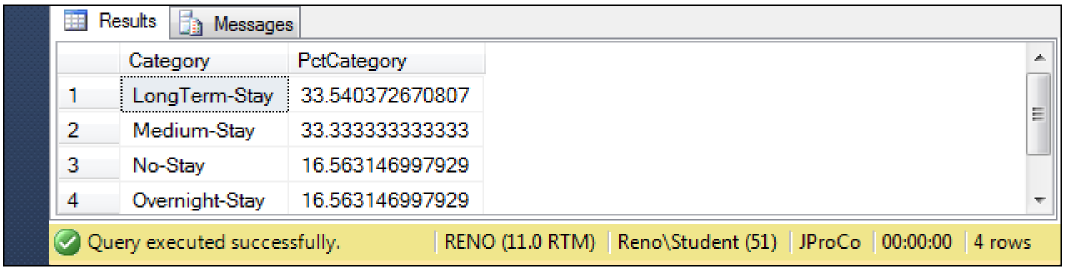


	ProductName	RetailPrice	Category	AvgPrice	AvgCatPrice
1	Underwater Tour 1 Week West Coast	307.415	LongTerm-Stay	316.671	559.0931
2	Underwater Tour 2 Weeks West Coast	553.347	LongTerm-Stay	316.671	559.0931
3	Underwater Tour 1 Week East Coast	404.295	LongTerm-Stay	316.671	559.0931
4	Underwater Tour 2 Weeks East Coast	727.731	LongTerm-Stay	316.671	559.0931
5	Underwater Tour 1 Week Mexico	525.295	LongTerm-Stay	316.671	559.0931
6	Underwater Tour 2 Weeks Mexico	945.531	LongTerm-Stay	316.671	559.0931

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 483 rows

Figure 5.27 Skill Check 2 adds a new expression field showing AvgCatPrice.

Skill Check 3: Use the JProCo database and query the CurrentProducts table. Show each distinct category and calculate the percentage (with decimals) of products for each Category. Since we have more LongTerm-Stay products, that category will represent the highest percentage of the total (Figure 5.28).

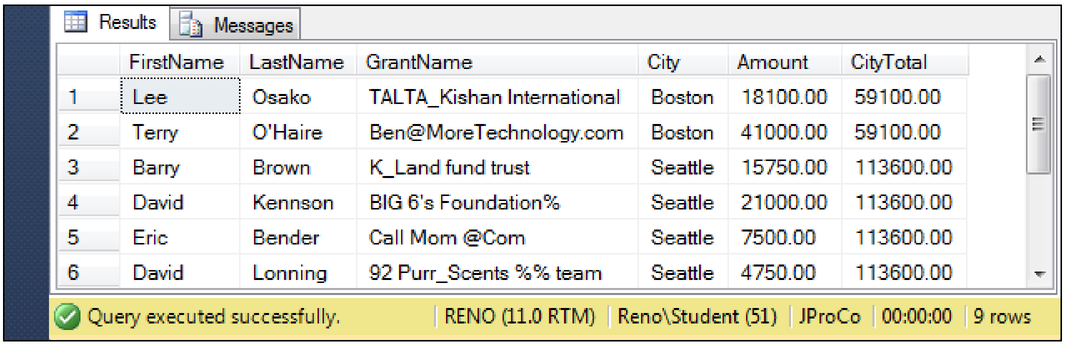


	Category	PctCategory
1	LongTerm-Stay	33.540372670807
2	Medium-Stay	33.333333333333
3	No-Stay	16.563146997929
4	Overnight-Stay	16.563146997929

```
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 4 rows
```

Figure 5.28 Skill Check 3 calculates the percentage of products in each product Category.

Skill Check 4: Join the Location, Employee and Grant tables and display FirstName, LastName, GrantName, City and Amount. Add an expression field called CityTotal that compares each grant to the total amount in the same City (See the expected result in Figure 5.29).



	FirstName	LastName	GrantName	City	Amount	CityTotal
1	Lee	Osako	TALTA_Kishan International	Boston	18100.00	59100.00
2	Terry	O'Haire	Ben@MoreTechnology.com	Boston	41000.00	59100.00
3	Barry	Brown	K_Land fund trust	Seattle	15750.00	113600.00
4	David	Kennson	BIG 6's Foundation%	Seattle	21000.00	113600.00
5	Eric	Bender	Call Mom @Com	Seattle	7500.00	113600.00
6	David	Lonning	92 Purr_Scents %% team	Seattle	4750.00	113600.00

```
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 9 rows
```

Figure 5.29 Compare each individual grant to the total grant amounts in the same city (CityTotal).

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab5.2_TheOVER_Clause.sql.

Lab 6.1: TOP(*n*) Queries

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter6.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

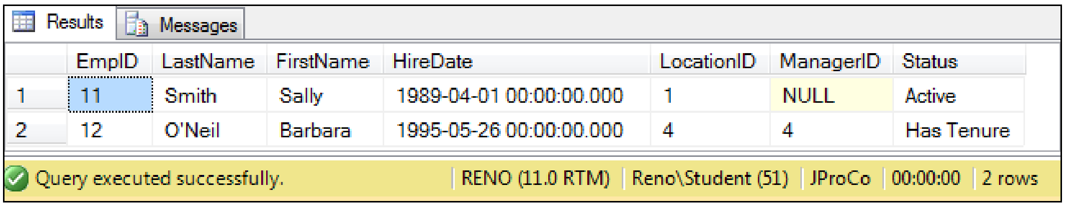
READER NOTE: *Some of the Skill Checks in Lab 6.1 will not work properly unless there are 81 records with 'ToBeDeleted = 1' in the CurrentProducts table.*

Before starting the first Skill Check, run the following query filtering on this criterion to verify 81 records are present.

```
SELECT *
FROM CurrentProducts
WHERE ToBeDeleted = 1
```

Skill Check 1: In the JProCo database, display only the two EmpID records with the oldest HireDate in the Employee table.

When done, the result set should resemble the one shown in Figure 6.18.



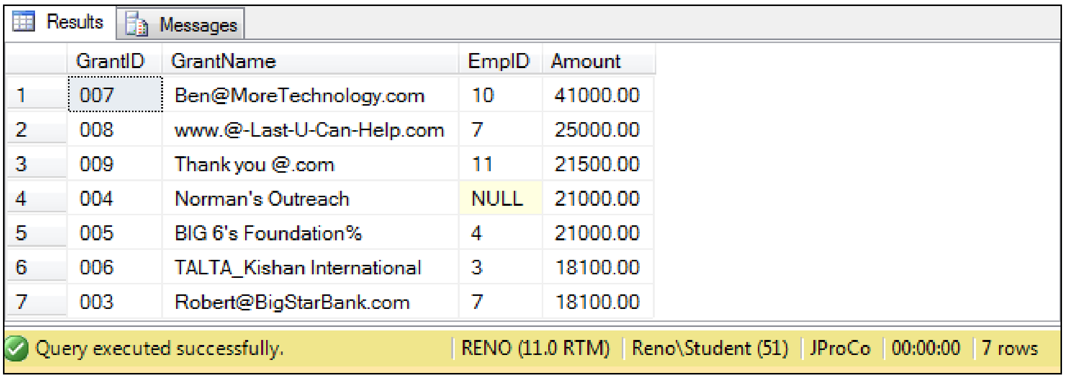
	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	11	Smith	Sally	1989-04-01 00:00:00.000	1	NULL	Active
2	12	O'Neil	Barbara	1995-05-26 00:00:00.000	4	4	Has Tenure

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 2 rows
```

Figure 6.18 Skill Check 1 finds the two employees with the oldest HireDate.

Skill Check 2: In the JProCo database, display the six largest grants found in the Grant table. Make sure any tied values will also appear in the result set.

When done, the result set should resemble the one shown in Figure 6.19.



Results | Messages

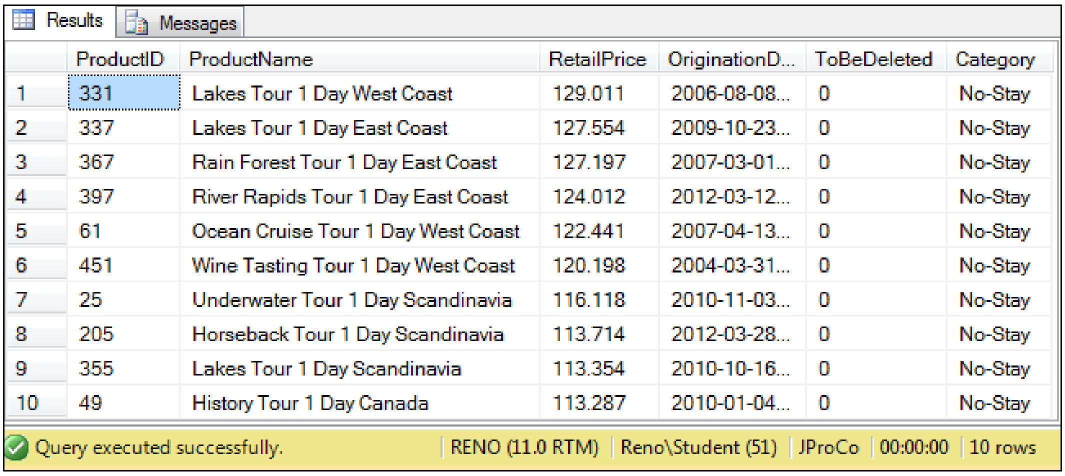
	GrantID	GrantName	EmpID	Amount
1	007	Ben@MoreTechnology.com	10	41000.00
2	008	www.@-Last-U-Can-Help.com	7	25000.00
3	009	Thank you @.com	11	21500.00
4	004	Norman's Outreach	NULL	21000.00
5	005	BIG 6's Foundation%	4	21000.00
6	006	TALTA_Kishan International	3	18100.00
7	003	Robert@BigStarBank.com	7	18100.00

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 7 rows

Figure 6.19 Skill Check 2 finds the top six grant values, including any ties for fourth and sixth place.

Skill Check 3: In the JProCo database, display the ten most expensive single day trips found in the CurrentProducts table. Since an overnight stay is not required, a day trip has a value of No-Stay in the Category field.

When done, the results should resemble those shown in Figure 6.20.



Results | Messages

	ProductID	ProductName	RetailPrice	OriginationD...	ToBeDeleted	Category
1	331	Lakes Tour 1 Day West Coast	129.011	2006-08-08...	0	No-Stay
2	337	Lakes Tour 1 Day East Coast	127.554	2009-10-23...	0	No-Stay
3	367	Rain Forest Tour 1 Day East Coast	127.197	2007-03-01...	0	No-Stay
4	397	River Rapids Tour 1 Day East Coast	124.012	2012-03-12...	0	No-Stay
5	61	Ocean Cruise Tour 1 Day West Coast	122.441	2007-04-13...	0	No-Stay
6	451	Wine Tasting Tour 1 Day West Coast	120.198	2004-03-31...	0	No-Stay
7	25	Underwater Tour 1 Day Scandinavia	116.118	2010-11-03...	0	No-Stay
8	205	Horseback Tour 1 Day Scandinavia	113.714	2012-03-28...	0	No-Stay
9	355	Lakes Tour 1 Day Scandinavia	113.354	2010-10-16...	0	No-Stay
10	49	History Tour 1 Day Canada	113.287	2010-01-04...	0	No-Stay

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 10 rows

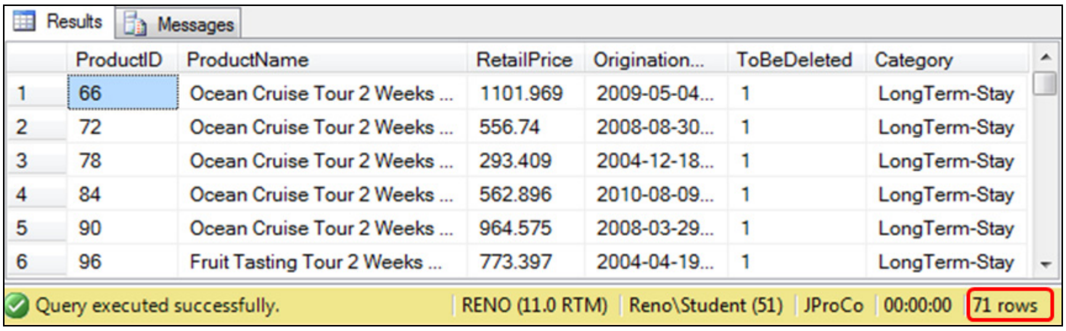
Figure 6.20 Skill Check 3 displays the top ten most expensive day trips.

Skill Check 4: Our sister company is now handling all trips lasting two weeks and thus need to be deleted from our CurrentProducts table. There are 81 records in the CurrentProducts table marked ToBeDeleted with a value of one (1).

Run a query to delete the first 10 records in the CurrentProducts table. Once this is complete, verify there are only 71 ToBeDeleted records with a value of one (1) remaining in the CurrentProducts table.

When done, the results should resemble those shown in Figure 6.21.

```
SELECT * FROM CurrentProducts
WHERE ToBeDeleted = 1
```



	ProductID	ProductName	RetailPrice	Origination...	ToBeDeleted	Category
1	66	Ocean Cruise Tour 2 Weeks ...	1101.969	2009-05-04...	1	LongTerm-Stay
2	72	Ocean Cruise Tour 2 Weeks ...	556.74	2008-08-30...	1	LongTerm-Stay
3	78	Ocean Cruise Tour 2 Weeks ...	293.409	2004-12-18...	1	LongTerm-Stay
4	84	Ocean Cruise Tour 2 Weeks ...	562.896	2010-08-09...	1	LongTerm-Stay
5	90	Ocean Cruise Tour 2 Weeks ...	964.575	2008-03-29...	1	LongTerm-Stay
6	96	Fruit Tasting Tour 2 Weeks ...	773.397	2004-04-19...	1	LongTerm-Stay

```
Query executed successfully.   RENO (11.0 RTM)   Reno\Student (51)   JProCo   00:00:00   71 rows
```

Figure 6.21 Confirm 71 ToBeDeleted records remain in the CurrentProducts table.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab6.1_Top_n_Queries.sql.

Lab 6.2: TOP(*n*) Tricks

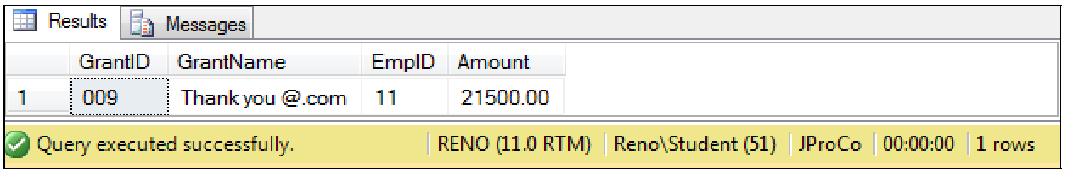
Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter6.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: In the JProCo database, write a query to find and then display only the third most expensive record from the Grant table based on the Amount field. ***Hint:*** *Locate the most expensive single GrantID that is not in the TOP(2) results.*

When done the results should resemble those shown in Figure 6.47.

```
SELECT TOP(1) *
--Remaining Code Here
```



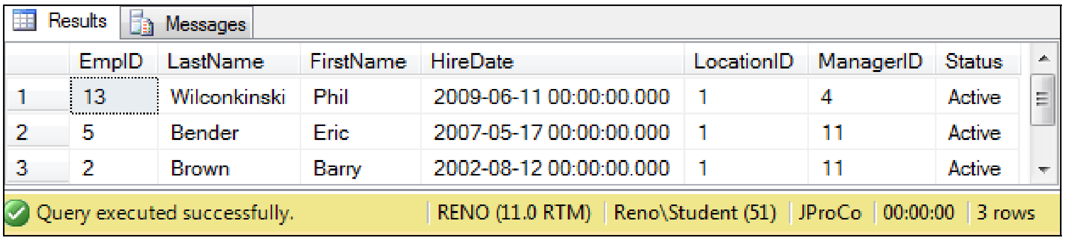
	GrantID	GrantName	EmpID	Amount
1	009	Thank you @.com	11	21500.00

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 1 rows
```

Figure 6.47 Skill Check 1 finds and displays the third most expensive record in the Grant table.

Skill Check 2: In the JProCo database context, write a query to find the three newest employees from the Employee table with a LocationID of 1.

When done, the results should resemble those shown in Figure 6.48.



	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	13	Wilconkinski	Phil	2009-06-11 00:00:00.000	1	4	Active
2	5	Bender	Eric	2007-05-17 00:00:00.000	1	11	Active
3	2	Brown	Barry	2002-08-12 00:00:00.000	1	11	Active

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 3 rows
```

Figure 6.48 Skill Check 2 shows the three most recently hired employees from LocationID 1.

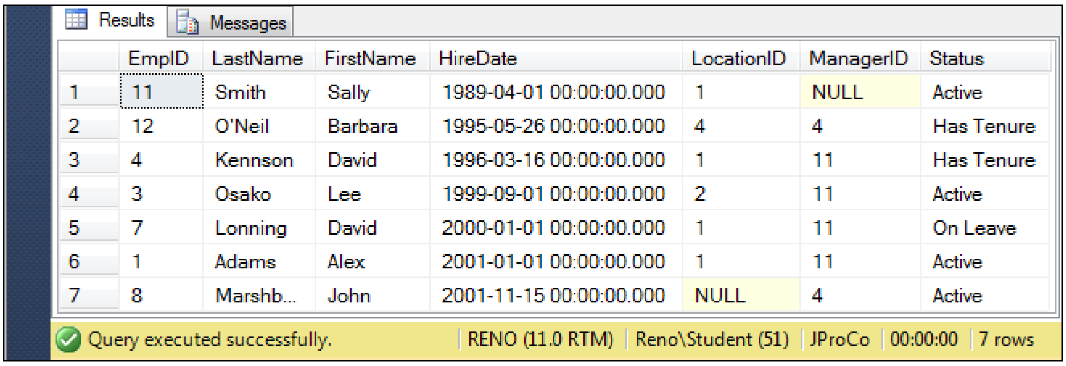
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab6.2_Top_n_Tricks.sql.

Lab 6.3: TOP(*n*) PERCENT Queries

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter6.3Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: From the Employee table of the JProCo database, show the top 50% most senior employees by HireDate.



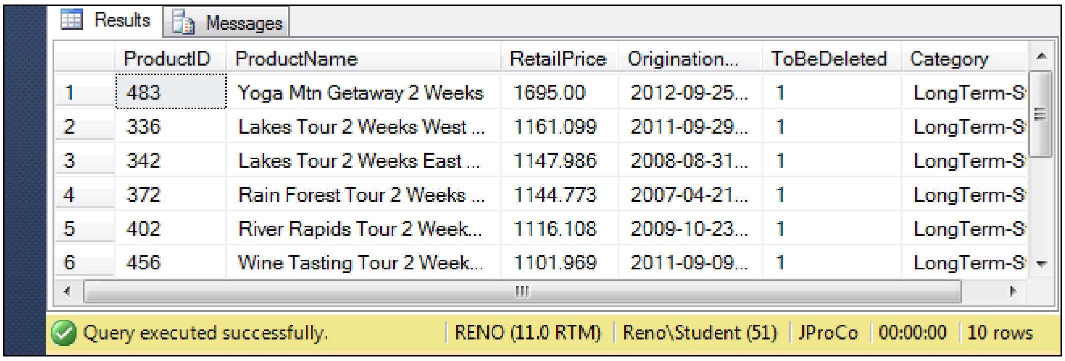
Results | Messages

	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	11	Smith	Sally	1989-04-01 00:00:00.000	1	NULL	Active
2	12	O'Neil	Barbara	1995-05-26 00:00:00.000	4	4	Has Tenure
3	4	Kennson	David	1996-03-16 00:00:00.000	1	11	Has Tenure
4	3	Osako	Lee	1999-09-01 00:00:00.000	2	11	Active
5	7	Lonning	David	2000-01-01 00:00:00.000	1	11	On Leave
6	1	Adams	Alex	2001-01-01 00:00:00.000	1	11	Active
7	8	Marshb...	John	2001-11-15 00:00:00.000	NULL	4	Active

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 7 rows

Figure 6.54 Skill Check 1 shows the top 50% of most senior employees.

Skill Check 2: From the CurrentProducts table of the JProCo database, show the top 2% most expensive products from the CurrentProducts table.



Results | Messages

	ProductID	ProductName	RetailPrice	Origination...	ToBeDeleted	Category
1	483	Yoga Mtn Getaway 2 Weeks	1695.00	2012-09-25...	1	LongTerm-S
2	336	Lakes Tour 2 Weeks West ...	1161.099	2011-09-29...	1	LongTerm-S
3	342	Lakes Tour 2 Weeks East ...	1147.986	2008-08-31...	1	LongTerm-S
4	372	Rain Forest Tour 2 Weeks ...	1144.773	2007-04-21...	1	LongTerm-S
5	402	River Rapids Tour 2 Week...	1116.108	2009-10-23...	1	LongTerm-S
6	456	Wine Tasting Tour 2 Week...	1101.969	2011-09-09...	1	LongTerm-S

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 10 rows

Figure 6.55 Skill Check 2 shows the top 2% most expensive products.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab6.3_Top_Percent_Queries.sql.

Lab 7.1: RANK() & DENSE_RANK()

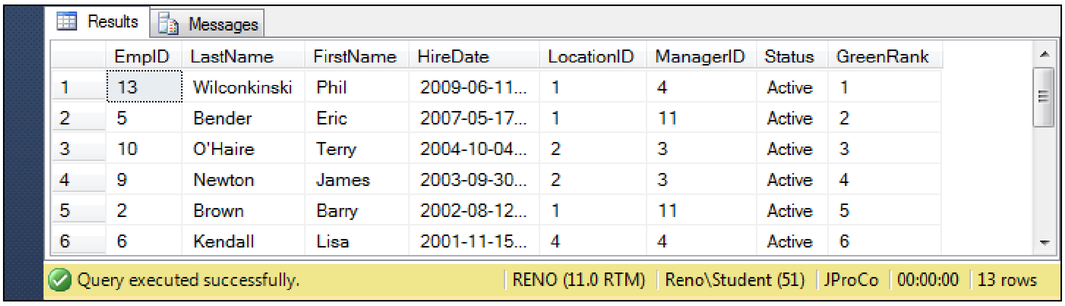
Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter7.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: In the JProCo database, use the correct ranking function to assign ranked values for each record in the Employee table based on the HireDate. The most recent HireDate should have a ranked value of 1 and each distinct date older than the first date should add 1 to the ranked value without any gaps.

The newest hires will appear at the top of the list and the oldest hire dates will be at the bottom of the list. The ranked field should be aliased as *GreenRank*.

When complete the results should resemble those shown in Figure 7.17.



	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status	GreenRank
1	13	Wilconkinski	Phil	2009-06-11...	1	4	Active	1
2	5	Bender	Eric	2007-05-17...	1	11	Active	2
3	10	O'Haire	Terry	2004-10-04...	2	3	Active	3
4	9	Newton	James	2003-09-30...	2	3	Active	4
5	2	Brown	Barry	2002-08-12...	1	11	Active	5
6	6	Kendall	Lisa	2001-11-15...	4	4	Active	6

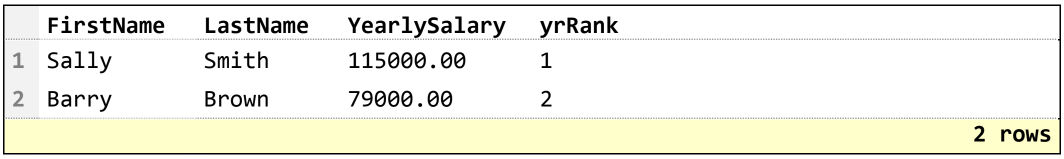
```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 13 rows
```

Figure 7.17 Skill Check 1 uses DENSE_RANK( ) to rank employees by the most recent HireDate.

Skill Check 2: In the JProCo database, join the Employee and PayRates tables together to display the FirstName, LastName, YearlySalary and a ranking expression aliased as *yrRank*. Order the ranked values based on the highest to lowest YearlySalary values.

The result set should only contain the fields for the first two ranked values provided by the RANK() function. ***Hint:*** *It is possible to complete this exercise without implementing a derived table when using the TOP(n) clause.*

When done, the results of the query should resemble those shown in Figure 7.18.



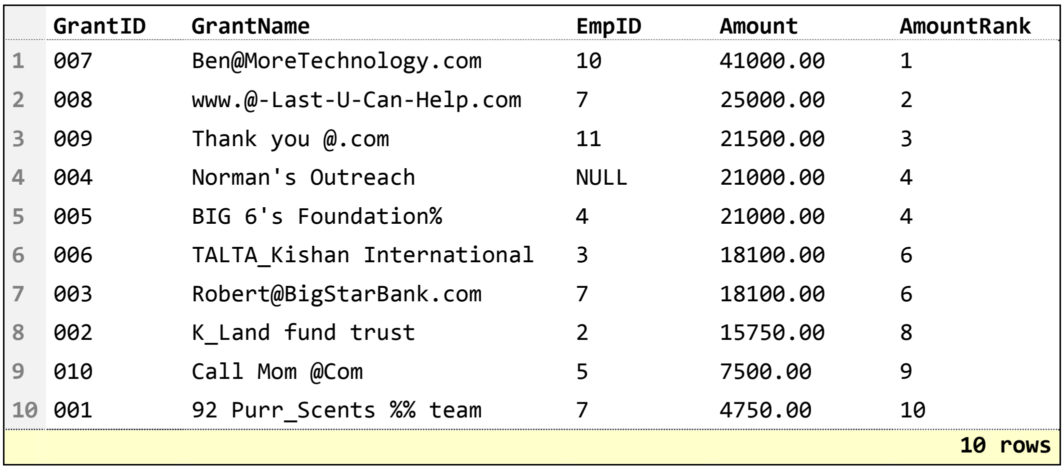
```
   FirstName   LastName   YearlySalary   yrRank
1  Sally       Smith      115000.00      1
2  Barry       Brown      79000.00       2
                                                          2 rows
```

Figure 7.18 The Employee and PayRates tables joined to display the two highest paid employees.

Skill Check 3: In the JProCo database, write a query that will display all the fields and records of the Grant table, plus a field to display the results of a ranking function that is aliased as *AmountRank*.

The ranking function should assign the ranked values based on sorting the Amount field from the highest to the lowest value. When encountering a tie, assign the same ranked value to each of the tied rows, allowing for gaps in the ranked values to occur when finding the next non-tied row.

When done, the query results should resemble those shown in Figure 7.19.



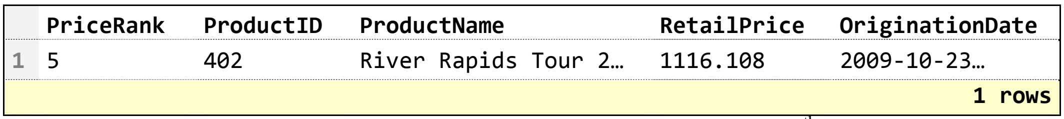
```
    GrantID   GrantName                     EmpID   Amount     AmountRank
1   007       Ben@MoreTechnology.com        10      41000.00   1
2   008       www.@-Last-U-Can-Help.com     7       25000.00   2
3   009       Thank you @.com               11      21500.00   3
4   004       Norman's Outreach             NULL    21000.00   4
5   005       BIG 6's Foundation%           4       21000.00   4
6   006       TALTA_Kishan International    3       18100.00   6
7   003       Robert@BigStarBank.com        7       18100.00   6
8   002       K_Land fund trust             2       15750.00   8
9   010       Call Mom @Com                 5       7500.00    9
10  001       92 Purr_Scents %% team        7       4750.00    10
                                                                 10 rows
```

Figure 7.19 Skill Check 3 ranks the Grant table showing ties and gaps.

Skill Check 4: In the JProCo database context, write a query that uses the DENSE_RANK() function to find only the 5th highest RetailPrice value in the CurrentProducts table. The expression field should be aliased as *PriceRank*. The field selection list should display only these fields in the following order: PriceRank, ProductID, ProductName, RetailPrice and Origination Date.

Hint: *Use a derived table to materialize the expression field, so it can be used as criteria to find the fifth record.*

When done the query results should resemble those shown in Figure 7.20.



```
   PriceRank   ProductID   ProductName            RetailPrice   OriginationDate
1  5           402         River Rapids Tour 2…   1116.108      2009-10-23…
                                                                        1 rows
```

Figure 7.20 Using DENSE_RANK() with a derived table to filter for the 5th ranked record.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab7.1_Rank&DenseRank.sql.

Lab 7.2: ROW_NUMBER

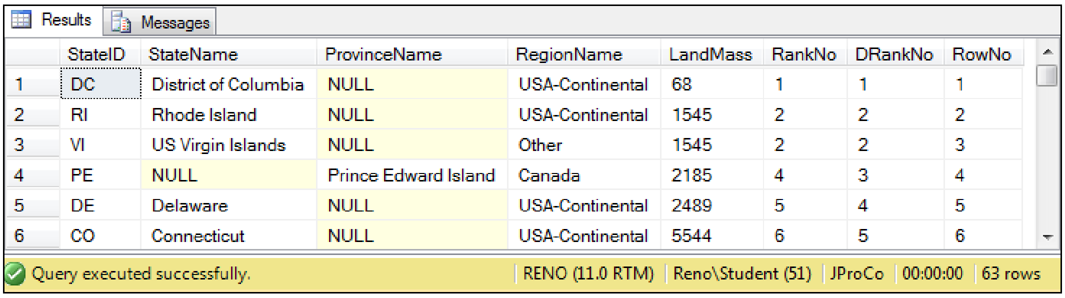
Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter7.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: In the JProCo database, locate the StateList table to write a query including all fields and records ranked by the smallest to largest value in the LandMass field.

In the SELECT list, add three columns named RankNo, DRankNo and RowNo (representing the RANK(), DENSE_RANK() and ROW_NUMBER() functions) after the four columns already included in the StateList table.

When done, the results of the query should resemble those shown in Figure 7.25.



	StateID	StateName	ProvinceName	RegionName	LandMass	RankNo	DRankNo	RowNo
1	DC	District of Columbia	NULL	USA-Continental	68	1	1	1
2	RI	Rhode Island	NULL	USA-Continental	1545	2	2	2
3	VI	US Virgin Islands	NULL	Other	1545	2	2	3
4	PE	NULL	Prince Edward Island	Canada	2185	4	3	4
5	DE	Delaware	NULL	USA-Continental	2489	5	4	5
6	CO	Connecticut	NULL	USA-Continental	5544	6	5	6

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 63 rows
```

Figure 7.25 Add three columns ranking the LandMass field of the StateList table.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab7.2_Row_Number.sql.

Lab 8.1: NTILE

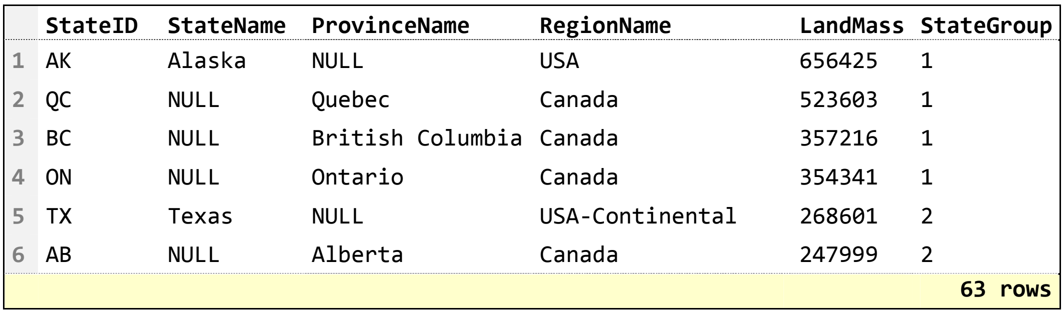
Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter8.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: In the JProCo database, locate the StateList table to write a query including all fields and records ranked by the largest to smallest value in the LandMass field. Group the rankings in 5% increments for all 63 records.

In the selection list, add a column named *StateGroup* (representing the NTILE() function) after the four columns already included in the StateList table.

When done, the query results should resemble those shown in Figure 8.7.



```
   StateID  StateName  ProvinceName      RegionName       LandMass  StateGroup
1  AK       Alaska     NULL              USA              656425    1
2  QC       NULL       Quebec            Canada           523603    1
3  BC       NULL       British Columbia  Canada           357216    1
4  ON       NULL       Ontario           Canada           354341    1
5  TX       Texas      NULL              USA-Continental  268601    2
6  AB       NULL       Alberta           Canada           247999    2
                                                                     63 rows
```

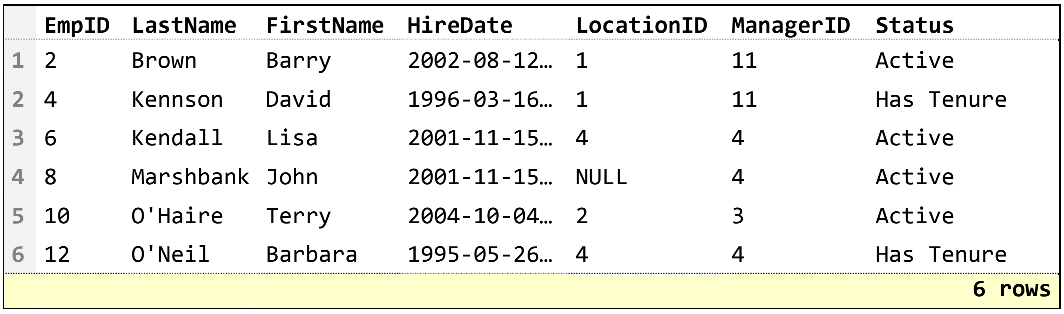
Figure 8.7 The StateList records grouped in 20 tiles using the NTILE( ) function.

Answer Code: The T-SQL code to this lab can be found in the downloadable files in a file named Lab8.1_NTILE.sql.

Lab 8.2: Predicating Row Functions

Lab Prep: Before we can begin the lab you must have SQL Server installed and have run the SQLQueries2012Vol2Chapter8.2Setup.sql script. When running a setup script, please first make sure you have closed all your query windows. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. One great way to do this is to close out of SQL Server, open SQL Server, and then run the script. Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Skill Check 1: Find even-numbered employees from the Employee table. When done your result should resemble Figure 8.28.

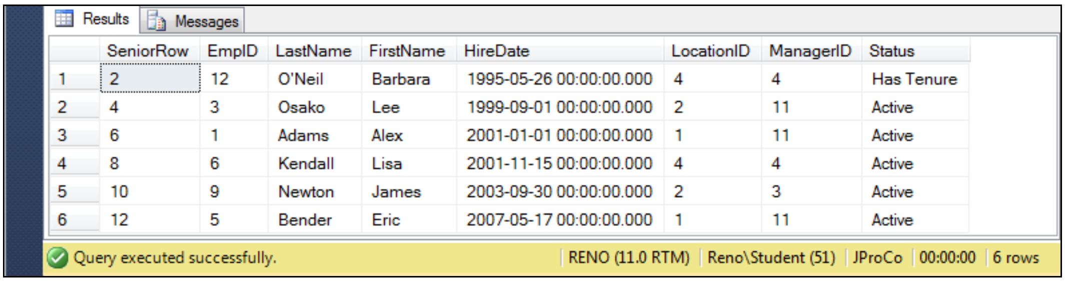


	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	2	Brown	Barry	2002-08-12…	1	11	Active
2	4	Kennson	David	1996-03-16…	1	11	Has Tenure
3	6	Kendall	Lisa	2001-11-15…	4	4	Active
4	8	Marshbank	John	2001-11-15…	NULL	4	Active
5	10	O'Haire	Terry	2004-10-04…	2	3	Active
6	12	O'Neil	Barbara	1995-05-26…	4	4	Has Tenure

6 rows

Figure 8.28 Skill Check 1.

Skill Check 2: From the Employee table of JProCo, show the even-numbered employee rows by HireDate. Alias the ranked row as SeniorRow. When done your result should resemble Figure 8.29.

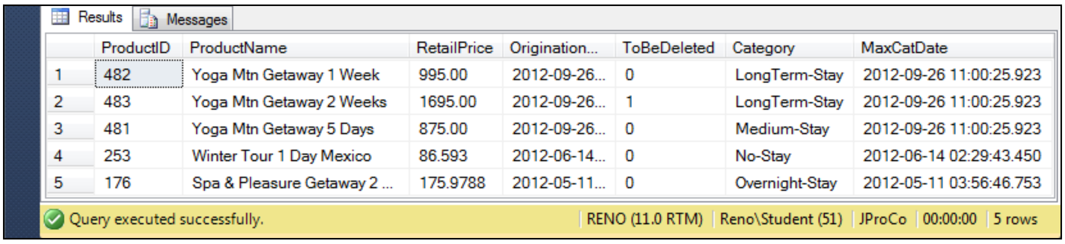


	SeniorRow	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	2	12	O'Neil	Barbara	1995-05-26 00:00:00.000	4	4	Has Tenure
2	4	3	Osako	Lee	1999-09-01 00:00:00.000	2	11	Active
3	6	1	Adams	Alex	2001-01-01 00:00:00.000	1	11	Active
4	8	6	Kendall	Lisa	2001-11-15 00:00:00.000	4	4	Active
5	10	9	Newton	James	2003-09-30 00:00:00.000	2	3	Active
6	12	5	Bender	Eric	2007-05-17 00:00:00.000	1	11	Active

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 6 rows

Figure 8.29 Skill Check 2.

Skill Check 3: From the CurrentProducts table of JProCo find the most recent product in each category. The expression field that finds the most recent origination date by category should be called MaxCatDate. When done your result should resemble Figure 8.30.


Results | Messages

	ProductID	ProductName	RetailPrice	Origination...	ToBeDeleted	Category	MaxCatDate
1	482	Yoga Mtn Getaway 1 Week	995.00	2012-09-26...	0	LongTerm-Stay	2012-09-26 11:00:25.923
2	483	Yoga Mtn Getaway 2 Weeks	1695.00	2012-09-26...	1	LongTerm-Stay	2012-09-26 11:00:25.923
3	481	Yoga Mtn Getaway 5 Days	875.00	2012-09-26...	0	Medium-Stay	2012-09-26 11:00:25.923
4	253	Winter Tour 1 Day Mexico	86.593	2012-06-14...	0	No-Stay	2012-06-14 02:29:43.450
5	176	Spa & Pleasure Getaway 2 ...	175.9788	2012-05-11...	0	Overnight-Stay	2012-05-11 03:56:46.753

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 5 rows
```

Figure 8.30 Skill Check 3.

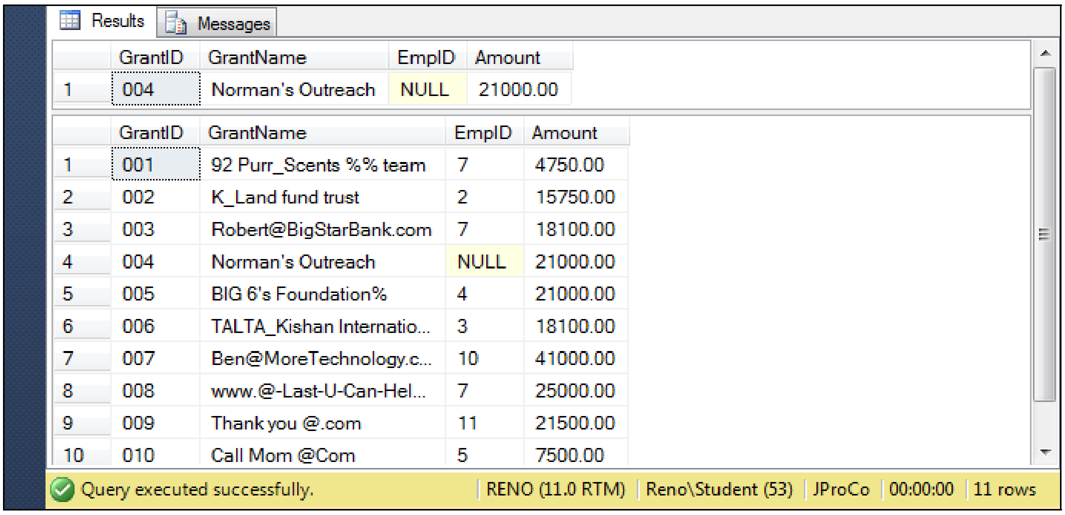
Answer Code: The T-SQL code to this lab can be found from the downloadable files named Lab8.2_ PredicatingRowFunctions.sql.

Lab 9.1: UNION and UNION ALL

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter9.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: The Grant table lists all grants, and vNonEmployeeGrants shows grants that were not found by an employee. Figure 9.18 shows these two result sets run at the same time. We can see they both include Norman's Outreach. This is an example of two different sets of data with similar metadata (in this case the metadata is the same).



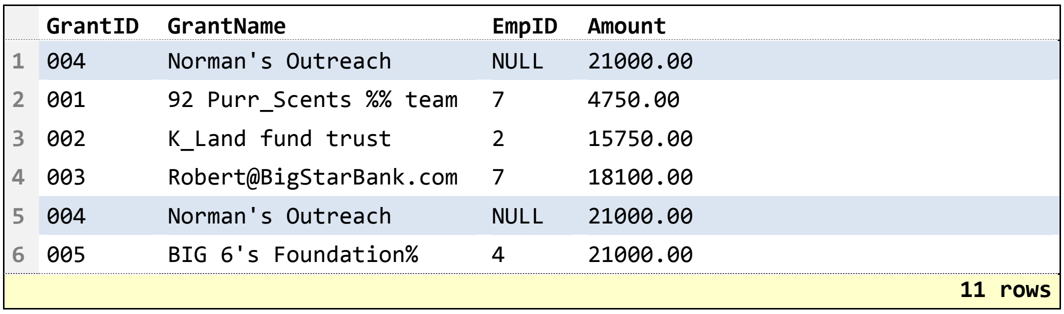
	GrantID	GrantName	EmpID	Amount
1	004	Norman's Outreach	NULL	21000.00

	GrantID	GrantName	EmpID	Amount
1	001	92 Purr_Scents %% team	7	4750.00
2	002	K_Land fund trust	2	15750.00
3	003	Robert@BigStarBank.com	7	18100.00
4	004	Norman's Outreach	NULL	21000.00
5	005	BIG 6's Foundation%	4	21000.00
6	006	TALTA_Kishan Internatio...	3	18100.00
7	007	Ben@MoreTechnology.c...	10	41000.00
8	008	www.@-Last-U-Can-Hel...	7	25000.00
9	009	Thank you @.com	11	21500.00
10	010	Call Mom @Com	5	7500.00

```
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 11 rows
```

Figure 9.18 Two queries with the same fields.

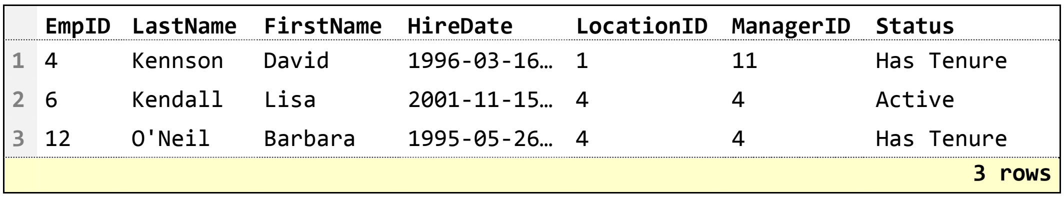
Write a query to combine the records from both tables. Since Norman's Outreach is found in both tables, it should appear in the result set twice. Use the correct set operator to achieve this result. When you're done, your result will look like Figure 9.19.



```
   GrantID  GrantName                EmpID  Amount
1  004      Norman's Outreach        NULL   21000.00
2  001      92 Purr_Scents %% team   7      4750.00
3  002      K_Land fund trust        2      15750.00
4  003      Robert@BigStarBank.com   7      18100.00
5  004      Norman's Outreach        NULL   21000.00
6  005      BIG 6's Foundation%      4      21000.00
                                                        11 rows
```

Figure 9.19 Combine records from two tables into one result set. Show Norman's Outreach twice.

Skill Check 2: You have two employees whose status shows they have received tenure. You also have two employees who work in Location 4. You have one employee working in Location 4 who has received tenure. Write two separate queries from the Employee table to find each group of employees. Then use the correct operator to put both result sets into one and show the distinct employees.



```
   EmpID  LastName  FirstName  HireDate      LocationID  ManagerID  Status
1  4      Kennson   David      1996-03-16…   1           11         Has Tenure
2  6      Kendall   Lisa       2001-11-15…   4           4          Active
3  12     O'Neil    Barbara    1995-05-26…   4           4          Has Tenure
                                                                        3 rows
```

Figure 9.20 Skill Check 2 shows all employees with tenure OR whose LocationID is 4.

When you are done, your result should resemble Figure 9.20.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab9.1_UnionAndUnionAll.sql.

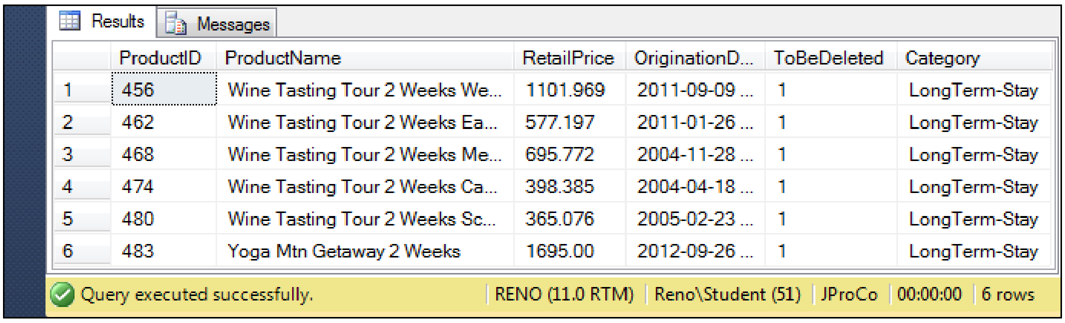
Lab 9.2: INTERSECT and EXCEPT

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter9.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: There are 81 records in your CurrentProducts table that are marked for deletion. Those products are to be moved to the RetiredProducts table. When you query the RetiredProducts table, notice there are only 75 records.

Use the correct set operator to find the six ToBeDeleted records from the CurrentProducts table that do not appear in the RetiredProducts table. When you are done, your result should resemble the figure you see here (Figure 9.29).



	ProductID	ProductName	RetailPrice	OriginationD...	ToBeDeleted	Category
1	456	Wine Tasting Tour 2 Weeks We...	1101.969	2011-09-09 ...	1	LongTerm-Stay
2	462	Wine Tasting Tour 2 Weeks Ea...	577.197	2011-01-26 ...	1	LongTerm-Stay
3	468	Wine Tasting Tour 2 Weeks Me...	695.772	2004-11-28 ...	1	LongTerm-Stay
4	474	Wine Tasting Tour 2 Weeks Ca...	398.385	2004-04-18 ...	1	LongTerm-Stay
5	480	Wine Tasting Tour 2 Weeks Sc...	365.076	2005-02-23 ...	1	LongTerm-Stay
6	483	Yoga Mtn Getaway 2 Weeks	1695.00	2012-09-26 ...	1	LongTerm-Stay

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 6 rows
```

Figure 9.29 Find the six records in the CurrentProducts table not contained in RetiredProducts.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab9.2_IntersectAndExcept.sql.

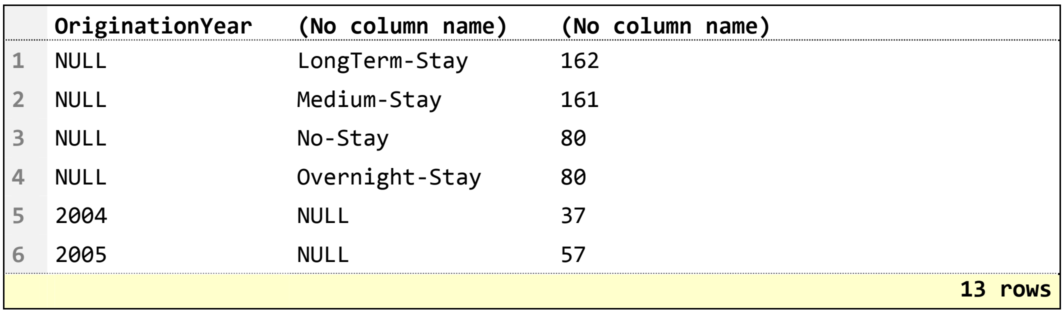
Lab 9.3: GROUPING SETS

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter9.3Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Turn this complicated UNION query into one that uses GROUPING SETS to achieve the same result.

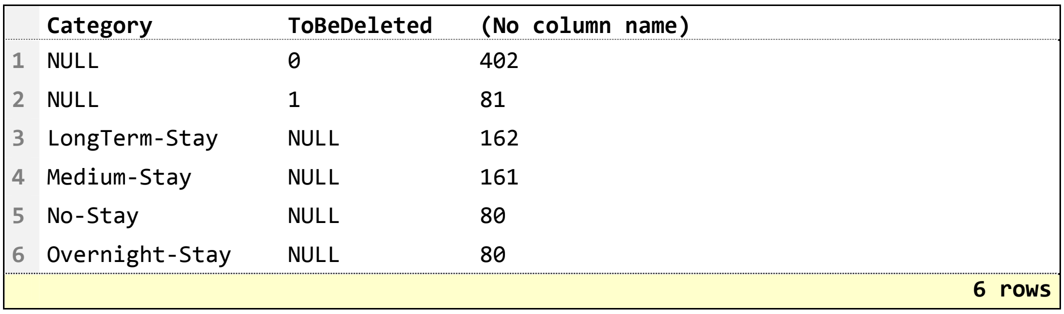
```
SELECT YEAR([OriginationDate]) AS OriginationYear,
NULL, COUNT(ProductID)
FROM CurrentProducts
GROUP BY YEAR([OriginationDate])

UNION

SELECT NULL AS OriginationYear, Category,
COUNT(ProductID)
FROM CurrentProducts
GROUP BY Category
```



```
   OriginationYear   (No column name)   (No column name)
1  NULL              LongTerm-Stay      162
2  NULL              Medium-Stay        161
3  NULL              No-Stay            80
4  NULL              Overnight-Stay     80
5  2004              NULL               37
6  2005              NULL               57
                                                 13 rows
```

Figure 9.49 Skill Check 1.

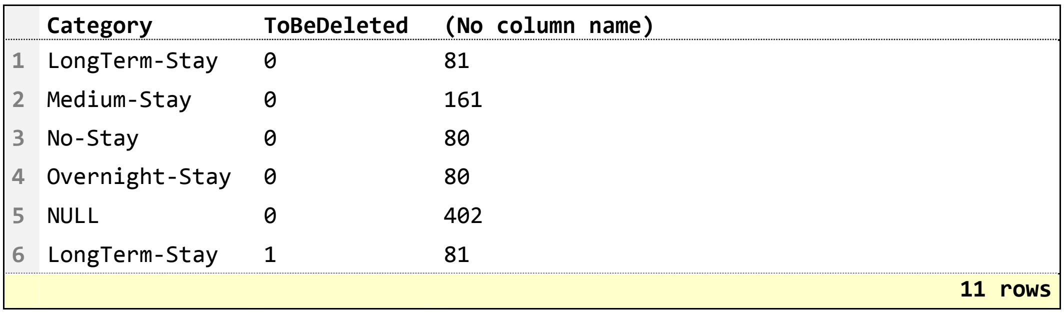
Skill Check 2: From the CurrentProducts table of JProCo, show the count of products by Category and ToBeDeleted.



```
  Category         ToBeDeleted    (No column name)
1 NULL             0              402
2 NULL             1              81
3 LongTerm-Stay    NULL           162
4 Medium-Stay      NULL           161
5 No-Stay          NULL           80
6 Overnight-Stay   NULL           80
                                                    6 rows
```

Figure 9.50 Skill Check 2.

Skill Check 3: From the CurrentProducts table of JProCo, show the count of products by Category and the count of ToBeDeleted products, as well as a composite of both fields as a grouped list set.



```
  Category         ToBeDeleted   (No column name)
1 LongTerm-Stay    0             81
2 Medium-Stay      0             161
3 No-Stay          0             80
4 Overnight-Stay   0             80
5 NULL             0             402
6 LongTerm-Stay    1             81
                                                   11 rows
```

Figure 9.51 Skill Check 3.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab9.3_GroupingSets.sql.

Lab 10.1: Common Table Expressions

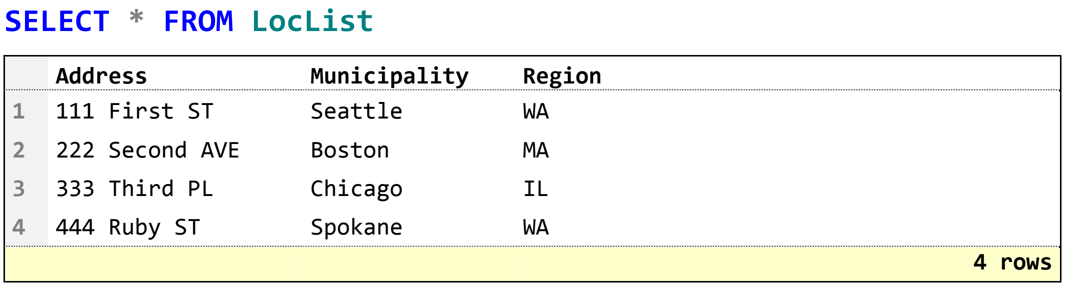
Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter10.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: You want to create an external report of your locations with a CTE called LocList. This list should show all fields of the Location table except for LocationID. The fields should also get new names, as shown in the following matrix.

- o Street should be shown as Address
- o City should be shown as Municipality
- o State should be shown as Region

When you're done, your result will resemble Figure 10.29. The CTE code will be just above the code you see in this figure.



```sql
SELECT * FROM LocList
```

```
   Address          Municipality    Region
1  111 First ST     Seattle         WA
2  222 Second AVE   Boston          MA
3  333 Third PL     Chicago         IL
4  444 Ruby ST      Spokane         WA
                                                    4 rows
```

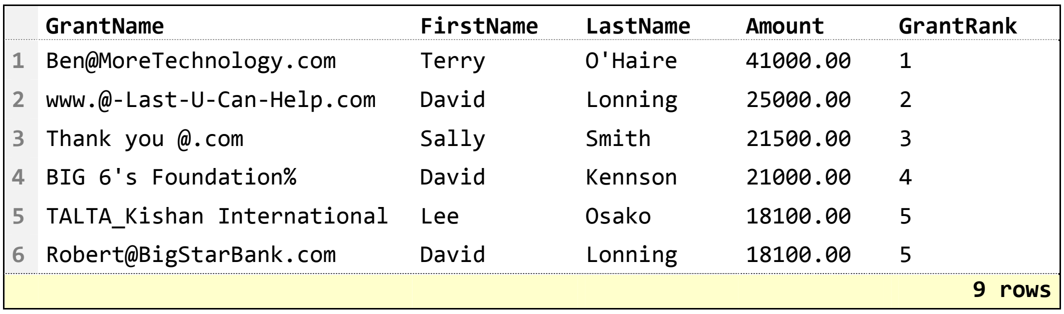
Figure 10.29 Skill Check 1 builds a CTE and aliases three fields from the Location table.

Skill Check 2: Create a CTE called EmpGrantRank that joins the Grant and Employee tables. You must be able to see the name of each employee who registered a grant. Include the following four fields in EmpGrantRank:

- GrantName
- FirstName
- LastName
- Amount

For your fifth field, create an expression field named GrantRank that uses the DENSE_RANK() function based on the amount field in descending order. When you're done, your result should resemble Figure 10.30.

READER NOTE: Amount will be in descending order.



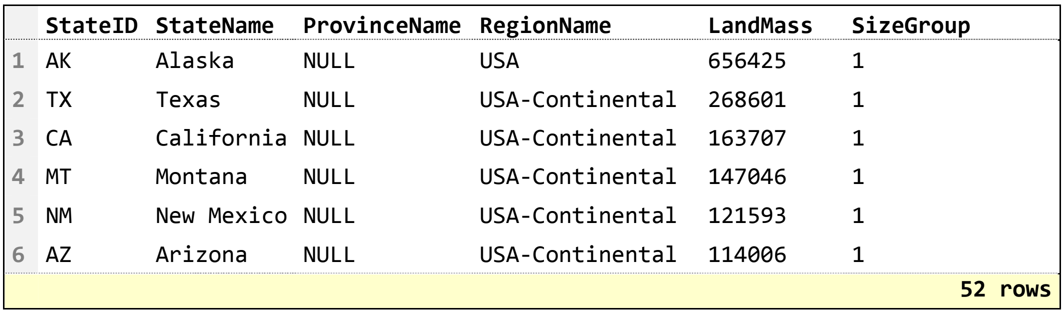
```
   GrantName                     FirstName  LastName  Amount    GrantRank
1  Ben@MoreTechnology.com        Terry      O'Haire   41000.00  1
2  www.@-Last-U-Can-Help.com     David      Lonning   25000.00  2
3  Thank you @.com               Sally      Smith     21500.00  3
4  BIG 6's Foundation%           David      Kennson   21000.00  4
5  TALTA_Kishan International    Lee        Osako     18100.00  5
6  Robert@BigStarBank.com        David      Lonning   18100.00  5
                                                                   9 rows
```

Figure 10.30 Write the CTE, GrantRank, with four fields plus a new expression field, GrantRank.

Skill Check 3: Create a CTE called StateRank that queries the StateList table and for all RegionName values that start with the letters USA. Add an expression field called SizeGroup that breaks down the landmass by size into 5 tiles. The top 20% should be in group 1; the next 20% should be in group 2 and the lowest 20% in group 5. When you're done, your result should resemble Figure 10.31.

READER NOTE: LandMass will be in descending order.

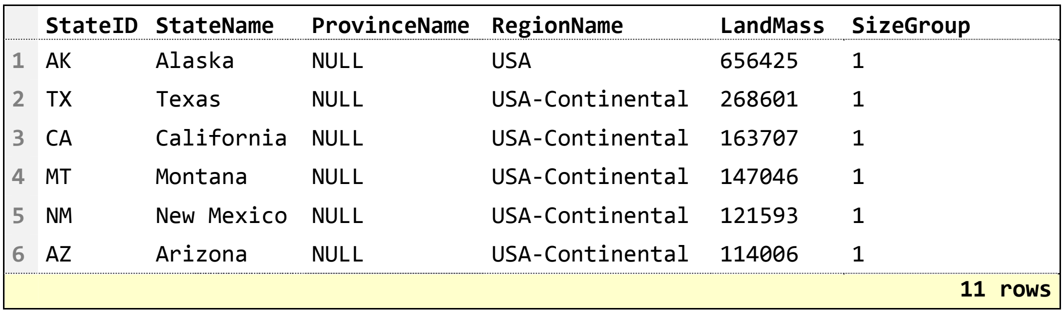
```sql
SELECT * FROM StateRank
```



```
   StateID  StateName   ProvinceName  RegionName      LandMass  SizeGroup
1  AK       Alaska      NULL          USA             656425    1
2  TX       Texas       NULL          USA-Continental 268601    1
3  CA       California  NULL          USA-Continental 163707    1
4  MT       Montana     NULL          USA-Continental 147046    1
5  NM       New Mexico  NULL          USA-Continental 121593    1
6  AZ       Arizona     NULL          USA-Continental 114006    1
                                                                  52 rows
```

Figure 10.31 Skill Check 3.

Skill Check 4: Take the result from Skill Check 3 and change the predicate so that the first group shows the top 20% largest states. You should get 11 records as seen in Figure 10.32.



```
  StateID StateName  ProvinceName RegionName       LandMass SizeGroup
1 AK      Alaska     NULL         USA              656425   1
2 TX      Texas      NULL         USA-Continental  268601   1
3 CA      California NULL         USA-Continental  163707   1
4 MT      Montana    NULL         USA-Continental  147046   1
5 NM      New Mexico NULL         USA-Continental  121593   1
6 AZ      Arizona    NULL         USA-Continental  114006   1
                                                              11 rows
```

Figure 10.32 Skill Check 4.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab10.1_CommonTableExpressions.sql.

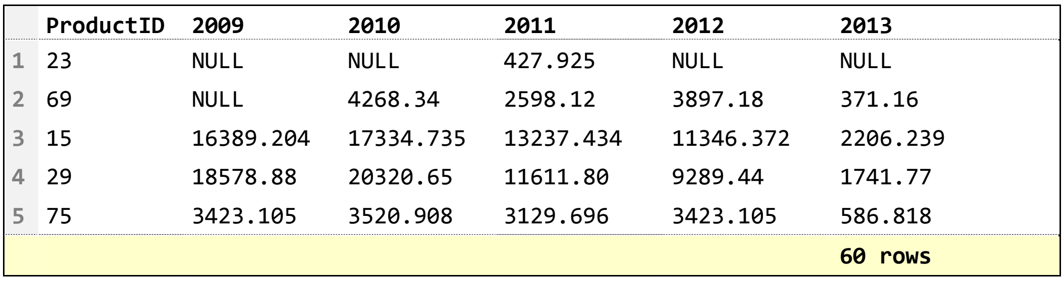
Lab 10.2: PIVOT

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter10.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Use PIVOT with the following aggregate query to achieve the result set you see in Figure 10.50.

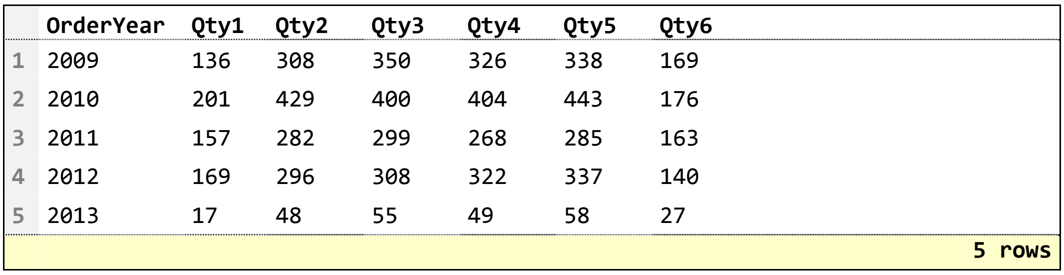
```
SELECT ProductID, OrderYear, RetailPrice
FROM vSales
```



```
   ProductID  2009        2010        2011        2012        2013
1  23         NULL        NULL        427.925     NULL        NULL
2  69         NULL        4268.34     2598.12     3897.18     371.16
3  15         16389.204   17334.735   13237.434   11346.372   2206.239
4  29         18578.88    20320.65    11611.80    9289.44     1741.77
5  75         3423.105    3520.908    3129.696    3423.105    586.818
                                                              60 rows
```

Figure 10.50 Skill Check 1.

Skill Check 2: 1627 unit quanties sold in 2009. 136 of thoes were purchased in a quantity of 1. 308 products were purchased in a quantity of 2 as you see in the first record. Knowing the quanity of sales per invoice from vSales, write a query using PIVOT and the dbo.vSales view to achieve the result set you see in Figure 10.51. Be sure to Alias the fields as seen here.



```
   OrderYear  Qty1  Qty2  Qty3  Qty4  Qty5  Qty6
1  2009       136   308   350   326   338   169
2  2010       201   429   400   404   443   176
3  2011       157   282   299   268   285   163
4  2012       169   296   308   322   337   140
5  2013       17    48    55    49    58    27
                                                    5 rows
```

Figure 10.51 Skill Check 2.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab10.2_PIVOT.sql.

Lab 10.3: UNPIVOT

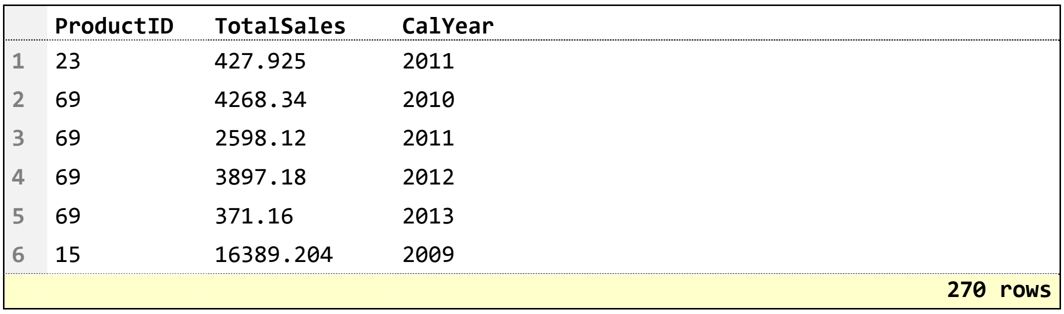
Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter10SpecialSetup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: After you run the SQLQueries2012Vol2Chapter10SpecialSetup.sql script then JProCo database will have the SalesGrid table.

```
SELECT *
FROM SalesGrid
```

Write a query using UNPIVOT which will achieve the result set you see in Figure 10.57.



```
   ProductID   TotalSales   CalYear
1  23          427.925      2011
2  69          4268.34      2010
3  69          2598.12      2011
4  69          3897.18      2012
5  69          371.16       2013
6  15          16389.204    2009
                                          270 rows
```

Figure 10.57 Skill Check 1.

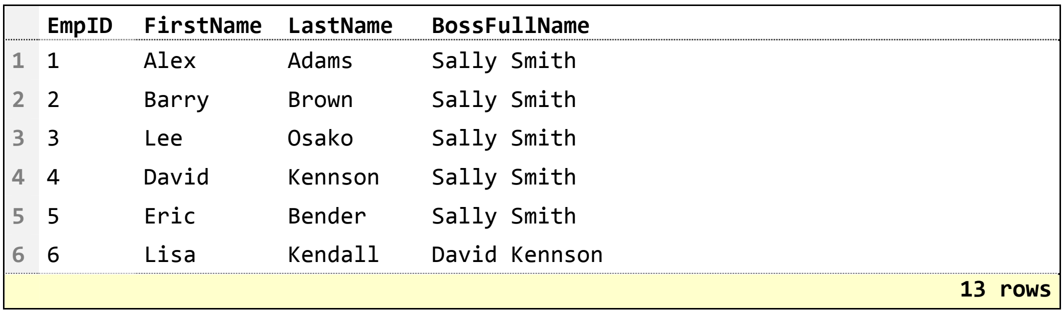
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab10.3_UNPIVOT.sql.

Lab 11.1: Self-Join Hierarchies

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter11.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: In JProCo join the Employee table to itself and just show each employee's EmpID, FirstName and LastName. Show each boss's first and last name as an expression field called *BossFullName*. The expression field should have a space in the middle. Also make sure to use the right type of outer join so that Sally Smith appears, even though she has no boss. When you are done, your result should resemble Figure 11.10.



```
   EmpID  FirstName  LastName  BossFullName
1  1      Alex       Adams     Sally Smith
2  2      Barry      Brown     Sally Smith
3  3      Lee        Osako     Sally Smith
4  4      David      Kennson   Sally Smith
5  5      Eric       Bender    Sally Smith
6  6      Lisa       Kendall   David Kennson
                                              13 rows
```

Figure 11.10 Skill Check 1 uses a self-join and adds the new expression field BossFullName.

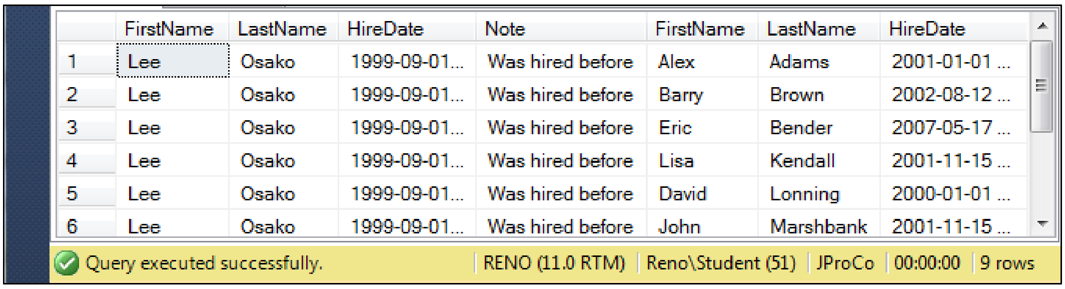
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab11.1_SelfJoins.sql.

Lab 11.2: Range Hierarchies

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter11.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: From the Employee table in JProCo write a query to see employee # 3 (Lee Osako) and all the people who were hired after him. Show just the FirstName, LastName and HireDate from both tables in the join. Make the fourth field an expression field that says "Was hired before". When you're done, your result will look like the figure you see here (Figure 11.23).

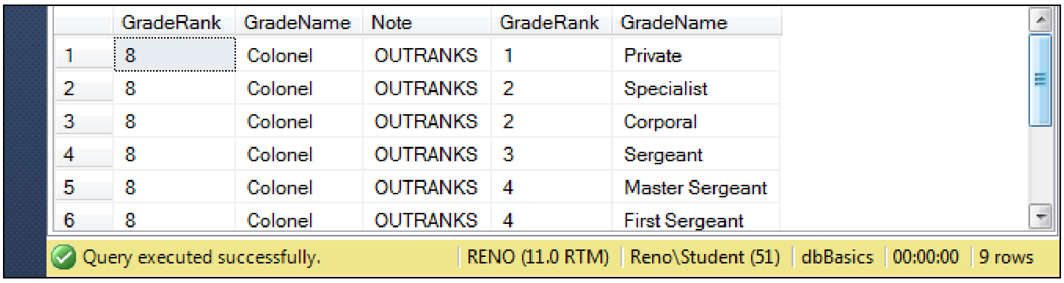


	FirstName	LastName	HireDate	Note	FirstName	LastName	HireDate
1	Lee	Osako	1999-09-01...	Was hired before	Alex	Adams	2001-01-01 ...
2	Lee	Osako	1999-09-01...	Was hired before	Barry	Brown	2002-08-12 ...
3	Lee	Osako	1999-09-01...	Was hired before	Eric	Bender	2007-05-17 ...
4	Lee	Osako	1999-09-01...	Was hired before	Lisa	Kendall	2001-11-15 ...
5	Lee	Osako	1999-09-01...	Was hired before	David	Lonning	2000-01-01 ...
6	Lee	Osako	1999-09-01...	Was hired before	John	Marshbank	2001-11-15 ...

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 9 rows
```

Figure 11.23 Skill Check 1 uses a self-join of the Employee table.

Skill Check 2: In dbBasics, use the Military table to write a range hierarchy showing the highest Army GradeRank and all the GradeRanks which are below Colonel. When you're done, your result will look like Figure 11.24.



	GradeRank	GradeName	Note	GradeRank	GradeName
1	8	Colonel	OUTRANKS	1	Private
2	8	Colonel	OUTRANKS	2	Specialist
3	8	Colonel	OUTRANKS	2	Corporal
4	8	Colonel	OUTRANKS	3	Sergeant
5	8	Colonel	OUTRANKS	4	Master Sergeant
6	8	Colonel	OUTRANKS	4	First Sergeant

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | dbBasics | 00:00:00 | 9 rows
```

Figure 11.24 Skill Check 2 uses a self-join of the Military table in the dbBasics database.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab11.2_RangeHierarchies.sql.

Points to Ponder - Range Hierarchies

1. The ON clause of your query can use many types of operators, such as:

- ‘=’ Exact match
- ‘>’ Greater than but not equal to
- ‘<’ Less than but not equal to
- ‘!=’ Not equal to (every value but…)

2. If there is more than one match in the join, then you will get more records in the result set than the original table.

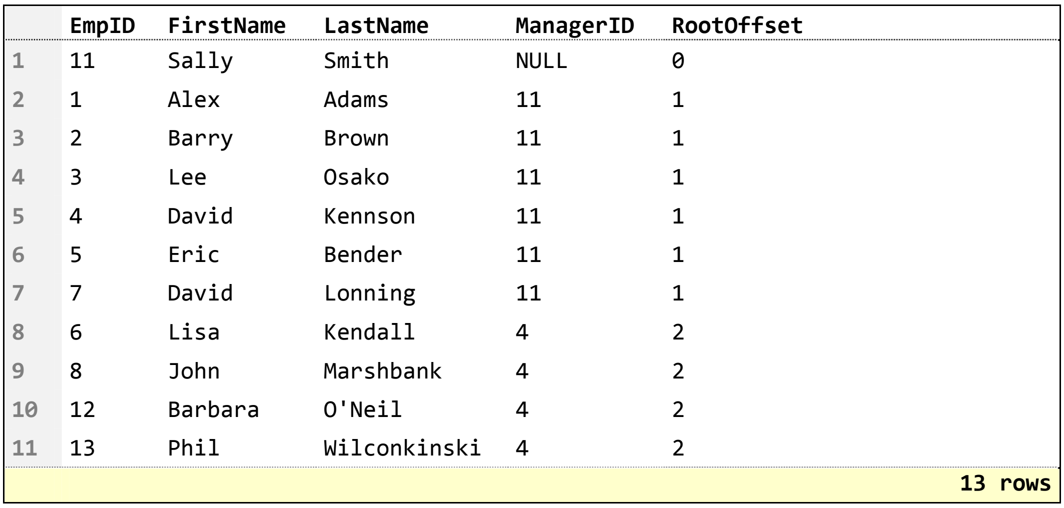
3. By using a ‘<’ or ‘>’ operator, we can show all results that compare to a particular record.

Lab 11.3: Recursive Queries

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter11.3Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Change the last query (Figure 11.33) so that you only measure everyone's level of distance from the CEO. For example Alex works directly for Sally, so he is just one level away from the root level of Sally Smith. She is the CEO, so her RootOffset would be zero. Modify the CTE query so that the highest level is 0 and the field is called RootOffset. When you're done, your result will resemble Figure 11.34



```
    EmpID  FirstName  LastName     ManagerID  RootOffset
1   11     Sally      Smith        NULL       0
2   1      Alex       Adams        11         1
3   2      Barry      Brown        11         1
4   3      Lee        Osako        11         1
5   4      David      Kennson      11         1
6   5      Eric       Bender       11         1
7   7      David      Lonning      11         1
8   6      Lisa       Kendall      4          2
9   8      John       Marshbank    4          2
10  12     Barbara    O'Neil       4          2
11  13     Phil       Wilconkinski 4          2
                                                  13 rows
```

Figure 11.34 You must modify the recursive CTE from our last example to show RootOffset.

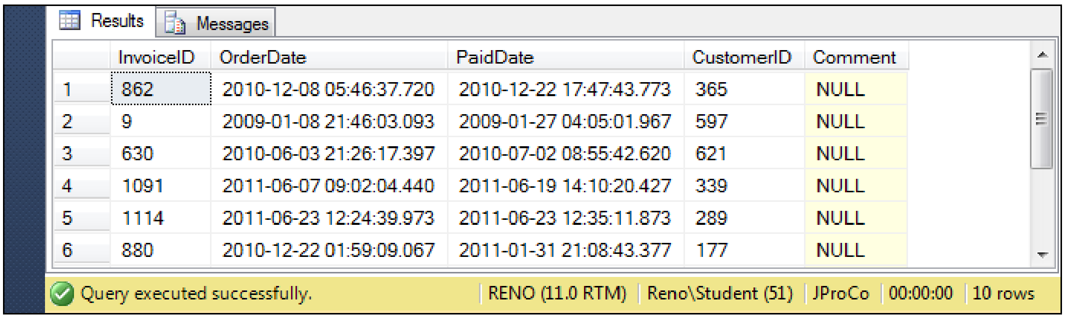
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab11.3_RecursiveQueries.sql.

Lab 12.1: Basic Subqueries

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter12.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Some invoices are large orders with many products on them. In the SalesInvoice table, 10 of the 1877 sales invoices contain more than 30 products. Run an aggregated subquery using the SalesInvoiceDetail table in order to find the InvoiceIDs for the 10 sales invoices containing more than 30 products. Feed those 10 InvoiceIDs into the criteria of the outer query. When you're done, your result will resemble the figure you see here (Figure 12.14).

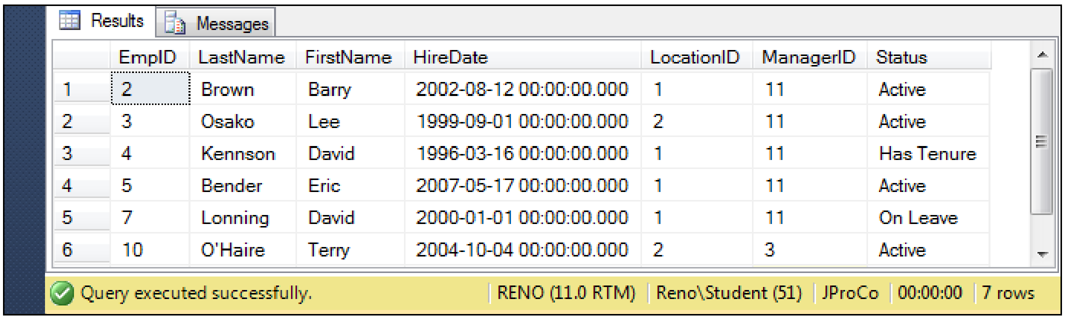


	InvoiceID	OrderDate	PaidDate	CustomerID	Comment
1	862	2010-12-08 05:46:37.720	2010-12-22 17:47:43.773	365	NULL
2	9	2009-01-08 21:46:03.093	2009-01-27 04:05:01.967	597	NULL
3	630	2010-06-03 21:26:17.397	2010-07-02 08:55:42.620	621	NULL
4	1091	2011-06-07 09:02:04.440	2011-06-19 14:10:20.427	339	NULL
5	1114	2011-06-23 12:24:39.973	2011-06-23 12:35:11.873	289	NULL
6	880	2010-12-22 01:59:09.067	2011-01-31 21:08:43.377	177	NULL

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 10 rows
```

Figure 12.14 Skill Check 1 finds the 10 sales invoices containing more than 30 products.

Skill Check 2: Write a subquery which will feed EmpIDs into an outer query of the Employee table. Show the records for just those employees who have found grants. When you're done, your result should resemble Figure 12.15.



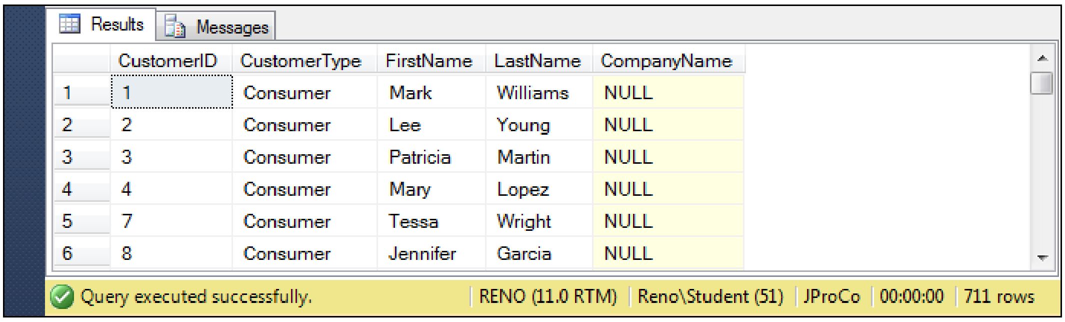
	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	2	Brown	Barry	2002-08-12 00:00:00.000	1	11	Active
2	3	Osako	Lee	1999-09-01 00:00:00.000	2	11	Active
3	4	Kennson	David	1996-03-16 00:00:00.000	1	11	Has Tenure
4	5	Bender	Eric	2007-05-17 00:00:00.000	1	11	Active
5	7	Lonning	David	2000-01-01 00:00:00.000	1	11	On Leave
6	10	O'Haire	Terry	2004-10-04 00:00:00.000	2	3	Active

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 7 rows
```

Figure 12.15 The employee records of the seven JProCo employees who found grants.

Skill Check 3: Query the Customer table using a subquery, which shows all the customers who have purchased (Hint: everyone appearing in the SalesInvoice table has bought something from JProCo). The query should show all customers who have ordered at least once from JProCo. If a customer has ordered multiple times, make sure they only show once in the result. When you're done, your result will resemble the figure you see here (Figure 12.16).

```
SELECT *
FROM Customer
WHERE CustomerID IN
-- Remaining Code Here
```


Results | Messages

	CustomerID	CustomerType	FirstName	LastName	CompanyName
1	1	Consumer	Mark	Williams	NULL
2	2	Consumer	Lee	Young	NULL
3	3	Consumer	Patricia	Martin	NULL
4	4	Consumer	Mary	Lopez	NULL
5	7	Consumer	Tessa	Wright	NULL
6	8	Consumer	Jennifer	Garcia	NULL

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 711 rows

Figure 12.16 The customer records of the 711 distinct customers who have bought from JProCo.

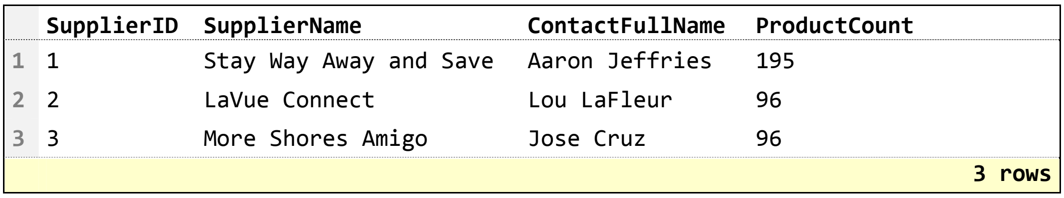
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab12.1_BasicSubqueries.sql.

Lab 12.2: Correlated Subqueries

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter12.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

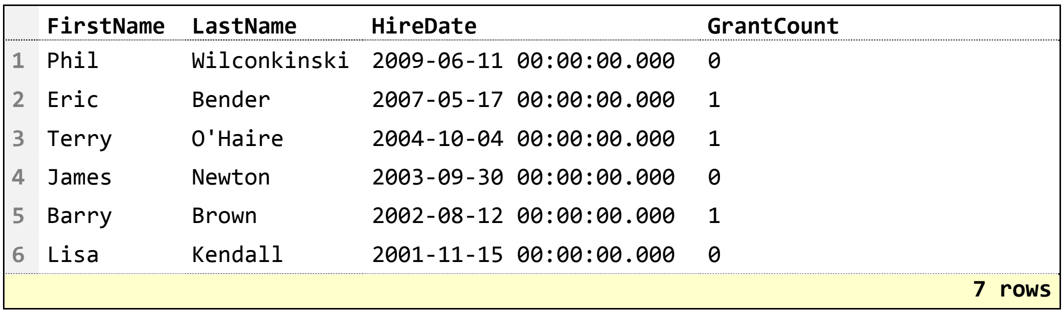
Skill Check 1: Query the Supplier table and use a subquery from the CurrentProducts table to create a ProductCount expression field. The ProductCount should show the number of products for each supplier. When you are done, your result should resemble the figure you see here (Figure 12.35).



```
   SupplierID  SupplierName            ContactFullName  ProductCount
1  1           Stay Way Away and Save  Aaron Jeffries   195
2  2           LaVue Connect           Lou LaFleur      96
3  3           More Shores Amigo       Jose Cruz        96
                                                                  3 rows
```

Figure 12.35 Skill Check 1 features a subquery, CurrentProducts. The outer query is Supplier.

Skill Check 2: Query the Employee table to find the 7 newest employees by hire date. Create an expression field called GrantCount that shows the number of grants found by each of those newest employees. Only the FirstName, LastName and HireDate columns and the expression field are to be shown. When you are done, your result should resemble the figure you see here (Figure 12.36).



```
   FirstName  LastName     HireDate                 GrantCount
1  Phil       Wilconkinski 2009-06-11 00:00:00.000  0
2  Eric       Bender       2007-05-17 00:00:00.000  1
3  Terry      O'Haire      2004-10-04 00:00:00.000  1
4  James      Newton       2003-09-30 00:00:00.000  0
5  Barry      Brown        2002-08-12 00:00:00.000  1
6  Lisa       Kendall      2001-11-15 00:00:00.000  0
                                                          7 rows
```

Figure 12.36 Skill Check 2 shows grant counts for JProCo's seven newest employees.

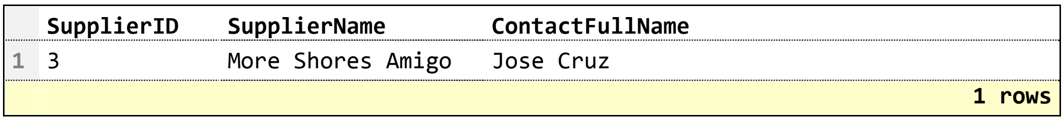
Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab12.2_CorrelatedSubqueries.sql.

Lab 12.3: Subquery Extensions

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter12.3Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

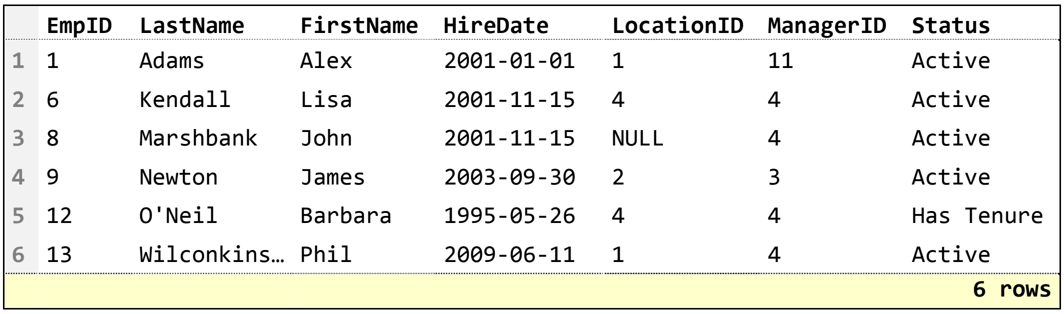
Skill Check 1: We want to compare the RetailPrice values of products supplied by JProCo's three external suppliers. Show the supplier who, if it were to introduce a $1000 product, that would represent the highest RetailPrice they offer. Find all suppliers (from the Supplier table) whose current products are less than $1000 (from the CurrentProducts table). Run the comparison using a correlated subquery (as we did in the last section examples) and use ANY, ALL, or SOME to modify your comparison operator. Use the CurrentProducts and Supplier tables in JProCo for your correlated subquery.



```
   SupplierID   SupplierName        ContactFullName
1  3            More Shores Amigo   Jose Cruz
                                                                    1 rows
```

Figure 12.50 Skill Check 1 uses a correlated subquery comparison with ANY, ALL, or SOME.

Skill Check 2: Show all employees who, if they were to find a $5000 grant, it would represent the highest grant they have ever found. Use the ALL operator with your correlated subquery. When you're done, your result will resemble the figure you see here (Figure 12.51).



```
   EmpID  LastName     FirstName  HireDate    LocationID  ManagerID  Status
1  1      Adams        Alex       2001-01-01  1           11         Active
2  6      Kendall      Lisa       2001-11-15  4           4          Active
3  8      Marshbank    John       2001-11-15  NULL        4          Active
4  9      Newton       James      2003-09-30  2           3          Active
5  12     O'Neil       Barbara    1995-05-26  4           4          Has Tenure
6  13     Wilconkins…  Phil       2009-06-11  1           4          Active
                                                                    6 rows
```

Figure 12.51 Skill Check 2 uses a correlated subquery between the Employee and Grant tables.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab12.3_SubqueryExtensions.sql.

Lab 13.1: Updates with Subqueries

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

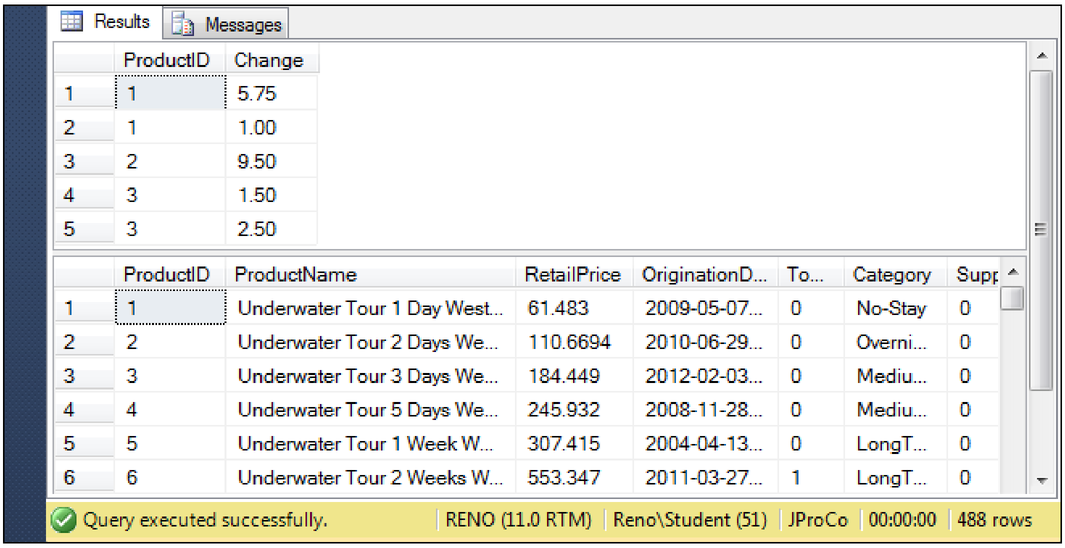
Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter13SpecialSetup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Review the data shown here in the PriceIncrease table. Turn the records of this table into a subquery which will increment prices in the CurrentProducts table.

***READER NOTE**: You should update only three records in the CurrentProducts table.*

```
SELECT * FROM PriceIncrease
```

```
SELECT * FROM CurrentProducts
```



	ProductID	Change
1	1	5.75
2	1	1.00
3	2	9.50
4	3	1.50
5	3	2.50

	ProductID	ProductName	RetailPrice	OriginationD...	To...	Category	Supp
1	1	Underwater Tour 1 Day West...	61.483	2009-05-07...	0	No-Stay	0
2	2	Underwater Tour 2 Days We...	110.6694	2010-06-29...	0	Overni...	0
3	3	Underwater Tour 3 Days We...	184.449	2012-02-03...	0	Mediu...	0
4	4	Underwater Tour 5 Days We...	245.932	2008-11-28...	0	Mediu...	0
5	5	Underwater Tour 1 Week W...	307.415	2004-04-13...	0	LongT...	0
6	6	Underwater Tour 2 Weeks W...	553.347	2011-03-27...	1	LongT...	0

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 488 rows

Figure 13.19 Your records before you start Skill Check 1.

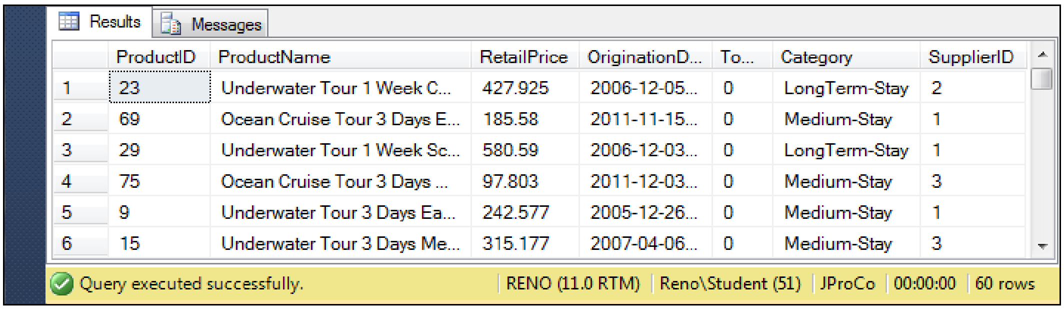
Lab 13.2: Existence Subqueries

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter13.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Use an EXISTS subquery to find all products that have been sold according to the SalesInvoiceDetail table.

```
SELECT *
FROM CurrentProducts AS cp
WHERE EXISTS (--Remaining Code Here)
```


Results | Messages

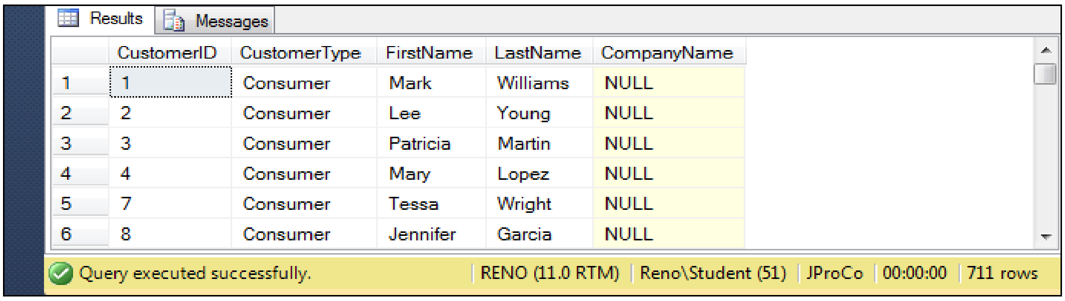
	ProductID	ProductName	RetailPrice	OriginationD...	To...	Category	SupplierID
1	23	Underwater Tour 1 Week C...	427.925	2006-12-05...	0	LongTerm-Stay	2
2	69	Ocean Cruise Tour 3 Days E...	185.58	2011-11-15...	0	Medium-Stay	1
3	29	Underwater Tour 1 Week Sc...	580.59	2006-12-03...	0	LongTerm-Stay	1
4	75	Ocean Cruise Tour 3 Days ...	97.803	2011-12-03...	0	Medium-Stay	3
5	9	Underwater Tour 3 Days Ea...	242.577	2005-12-26...	0	Medium-Stay	1
6	15	Underwater Tour 3 Days Me...	315.177	2007-04-06...	0	Medium-Stay	3

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 60 rows
```

Figure 13.33 Skill Check 1 finds all products that have been sold.

Skill Check 2: Use an EXISTS subquery to find all customers who have made a purchase according to the SalesInvoice table.

```
SELECT *
FROM Customer AS cu
WHERE EXISTS (--Remaining Code Here)
```



Results | Messages

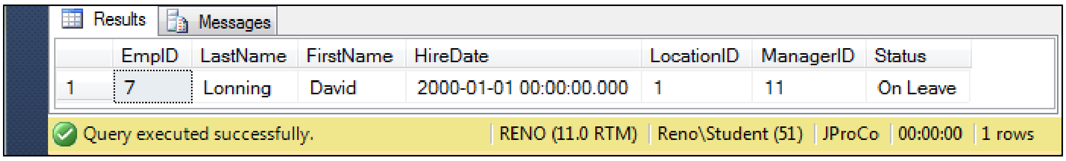
	CustomerID	CustomerType	FirstName	LastName	CompanyName
1	1	Consumer	Mark	Williams	NULL
2	2	Consumer	Lee	Young	NULL
3	3	Consumer	Patricia	Martin	NULL
4	4	Consumer	Mary	Lopez	NULL
5	7	Consumer	Tessa	Wright	NULL
6	8	Consumer	Jennifer	Garcia	NULL

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 711 rows
```

Figure 13.34 Skill Check 2 finds all customers who have made a purchase.

Skill Check 3: Use an EXISTS subquery to find all employees who have found more than two grants.

```
SELECT *
FROM Employee AS em
WHERE EXISTS (--Remaining Code Here)
```



Results | Messages

	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	7	Lonning	David	2000-01-01 00:00:00.000	1	11	On Leave

```
Query executed successfully.   RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 1 rows
```

Figure 13.35 Skill Check 3 finds employees that have procured at least three grants.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab13.2_ExistenceSubqueries.sql.

Lab 14.1: Using Merge

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

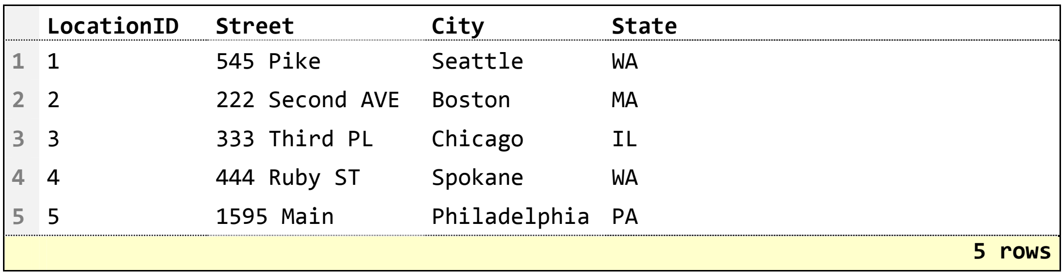
Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter14.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: Create a stored procedure called UpsertLocation that accepts four parameters – one for each field in the Location table. This stored procedure should make updates to existing records and insert any new records. After creating the stored procedure, call on it with the following code.

```
EXEC UpsertLocation 1,'545 Pike', 'Seattle', 'WA'

EXEC UpsertLocation 5,'1595 Main', 'Philadelphia', 'PA'
```

Check to see that the update was made to Location 1 and a new record (Location 5) was inserted by running a query on the Location table. When you are done, your result should resemble Figure 14.18.

```
SELECT * FROM Location
```



```
   LocationID  Street          City          State
1  1           545 Pike        Seattle       WA
2  2           222 Second AVE  Boston        MA
3  3           333 Third PL    Chicago       IL
4  4           444 Ruby ST     Spokane       WA
5  5           1595 Main       Philadelphia  PA
                                                                    5 rows
```

Figure 14.18 Skill Check 1 updates one record and inserts one record.

Skill Check 2: You have a table named PayRatesFeed with the updated pay information that needs to be fed into the PayRates table (Figure 14.19).

```
SELECT * FROM PayRatesFeed
```

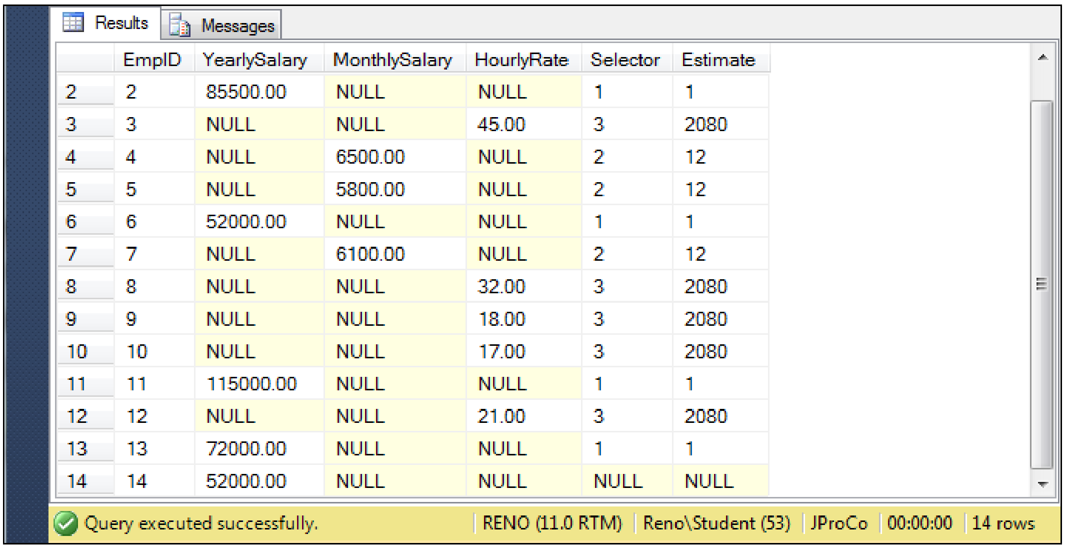
	EmpID	YearlySalary	MonthlySalary	HourlyRate	Selector	Estimate
1	1	97500.00	NULL	NULL	1	1
2	2	85500.00	NULL	NULL	1	1
3	14	52000.00	NULL	NULL	1	1

3 rows

Figure 14.19 Skill Check 2 uses PayRatesFeed as a source table for the target table, PayRates.

Write a MERGE statement that will update employee 1 to a YearlySalary of $97500 and insert a new pay record for EmpID 14 with year salary of $52000. When you are done, your result should resemble Figure 14.20.

```
SELECT * FROM PayRates
```



	EmpID	YearlySalary	MonthlySalary	HourlyRate	Selector	Estimate
2	2	85500.00	NULL	NULL	1	1
3	3	NULL	NULL	45.00	3	2080
4	4	NULL	6500.00	NULL	2	12
5	5	NULL	5800.00	NULL	2	12
6	6	52000.00	NULL	NULL	1	1
7	7	NULL	6100.00	NULL	2	12
8	8	NULL	NULL	32.00	3	2080
9	9	NULL	NULL	18.00	3	2080
10	10	NULL	NULL	17.00	3	2080
11	11	115000.00	NULL	NULL	1	1
12	12	NULL	NULL	21.00	3	2080
13	13	72000.00	NULL	NULL	1	1
14	14	52000.00	NULL	NULL	NULL	NULL

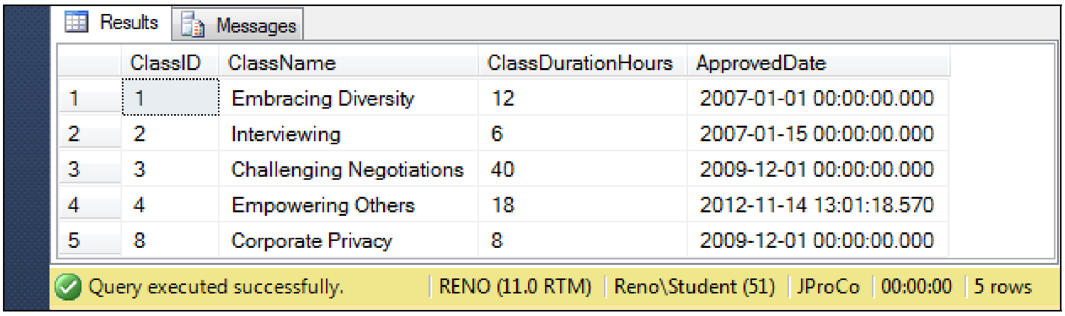
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (53) | JProCo | 00:00:00 | 14 rows

Figure 14.20 Your MERGE statement will update the target table with data from the source table.

Skill Check 3: Create a stored procedure called UpsertMgmtTraining that accepts four parameters for each field in the MgmtTraining table. This stored procedure should make updates to existing records and insert any new records. Create the stored procedure and call on it with the following code.

Check to see that the updates were made to Class 3 and a new class was inserted by running a query on the MgmtTraining table. When you are done, your result should resemble Figure 14.21.

```
EXEC UpsertMgmtTraining 3, 'Challenging Negotiations', 40,
'12/1/2009'

EXEC UpsertMgmtTraining 0, 'Corporate Privacy', 8,
'12/1/2009'

SELECT * FROM MgmtTraining
```



	ClassID	ClassName	ClassDurationHours	ApprovedDate
1	1	Embracing Diversity	12	2007-01-01 00:00:00.000
2	2	Interviewing	6	2007-01-15 00:00:00.000
3	3	Challenging Negotiations	40	2009-12-01 00:00:00.000
4	4	Empowering Others	18	2012-11-14 13:01:18.570
5	8	Corporate Privacy	8	2009-12-01 00:00:00.000

```
Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 5 rows
```

Figure 14.21 After creating and executing your sproc UpsertMgmtTraining, check your results.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab14.1_UsingMerge.sql

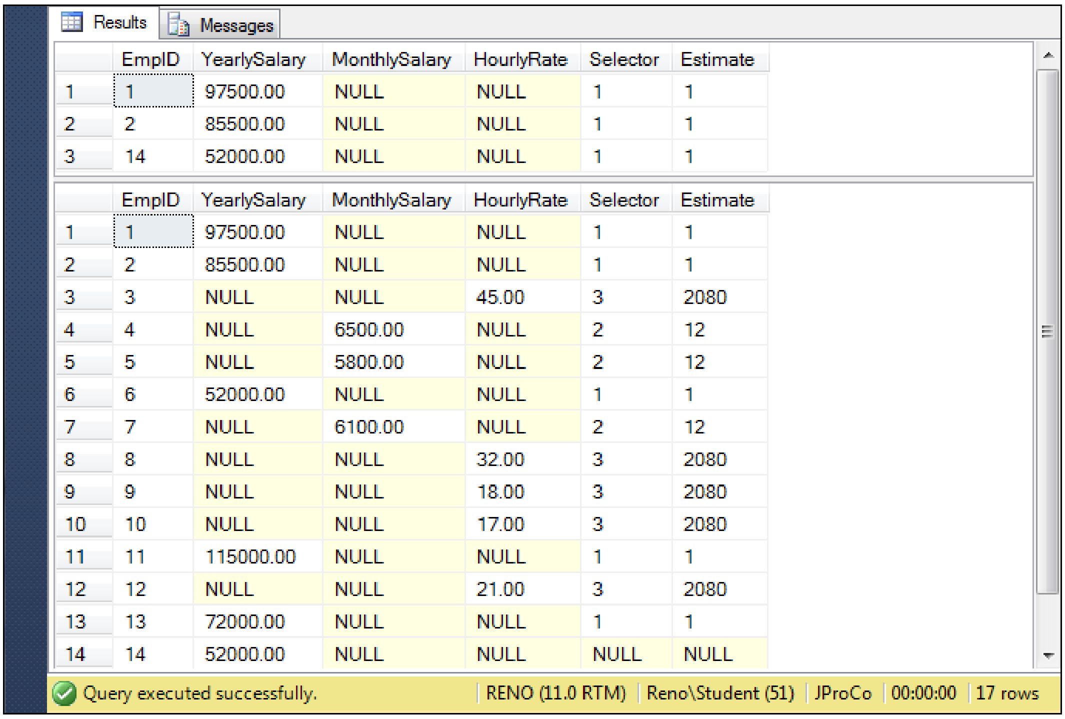
Lab 14.2: MERGE Updating Options

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter14.2Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: The PayRatesFeed table has three records. The first record shows a yearly salary of 97500 per year which is identical to employee 1's pay in the PayRates table. This record should not be updated. The PayRatesFeed table shows the new pay for employee 2 will be 85500 per year, which is higher than the current 79000 in the PayRates table. This record will need to be updated. The last record from the PayRatesFeed table shows a pay of 52000 per year for employee 14. Since there is no employee 14 in the PayRates table, you must insert this record. Write a MERGE statement that will insert new records and update existing records only if their YearlySalary has changed. When done, your result will look like Figure 14.35.

```
SELECT * FROM PayRatesFeed
SELECT * FROM PayRates
```



	EmpID	YearlySalary	MonthlySalary	HourlyRate	Selector	Estimate
1	1	97500.00	NULL	NULL	1	1
2	2	85500.00	NULL	NULL	1	1
3	14	52000.00	NULL	NULL	1	1

	EmpID	YearlySalary	MonthlySalary	HourlyRate	Selector	Estimate
1	1	97500.00	NULL	NULL	1	1
2	2	85500.00	NULL	NULL	1	1
3	3	NULL	NULL	45.00	3	2080
4	4	NULL	6500.00	NULL	2	12
5	5	NULL	5800.00	NULL	2	12
6	6	52000.00	NULL	NULL	1	1
7	7	NULL	6100.00	NULL	2	12
8	8	NULL	NULL	32.00	3	2080
9	9	NULL	NULL	18.00	3	2080
10	10	NULL	NULL	17.00	3	2080
11	11	115000.00	NULL	NULL	1	1
12	12	NULL	NULL	21.00	3	2080
13	13	72000.00	NULL	NULL	1	1
14	14	52000.00	NULL	NULL	NULL	NULL

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 17 rows
```

Figure 14.35 Write code to update one record and insert one record into the PayRates table.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab14.2_MergeUpdateOptions.sql

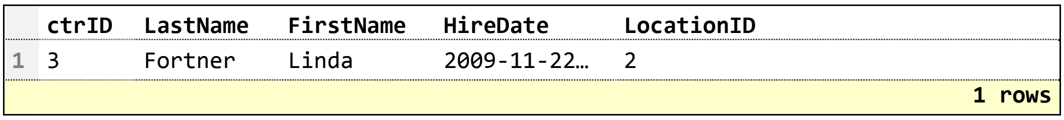
Lab 15.1: Using Output

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter15.1Setup.sql), please make sure to close all query windows within SSMS. An open query window pointing to a database context can lock that database preventing it from updating when the script is executing. A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

Skill Check 1: The Contractor table has four records: three records from Location 1 and one from Location 2. JProCo is closing down Location 2 next year and thus must delete one record from the Contractor table. As this record is being deleted, we want to see the affected record displayed in our result pane. Write this query and execute the deletion with the appropriate output statement. When you're done, the output will show on your screen (Figure 15.24).

```sql
DELETE FROM Contractor
--Remaining Code Here
```

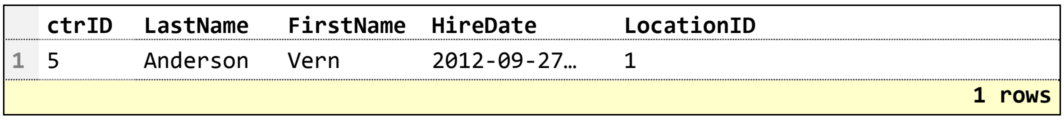


```
  ctrID  LastName  FirstName  HireDate       LocationID
1 3      Fortner   Linda      2009-11-22…    2
                                                                 1 rows
```

Figure 15.24 Delete Location 2 from the Contractor table. Use OUTPUT to display the change.

Skill Check 2: A new contractor named Vern Anderson is coming onboard to work in Location 1. Write the code to execute this insert, and use the GETDATE() function to populate the current date and time in the HireDate field of the Contractor table. Use the OUTPUT clause to show the results of your insertion as you run it. When you're done, your result should resemble Figure 15.25.

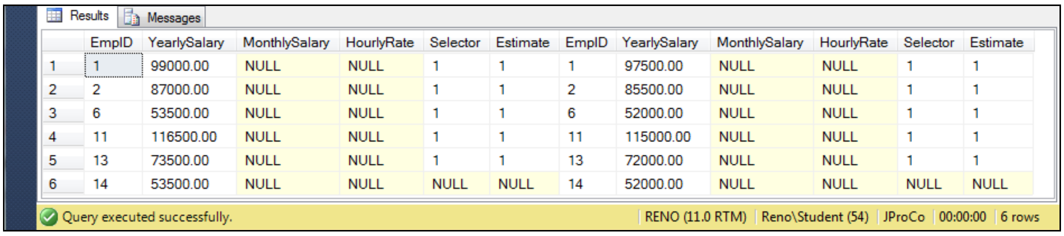
```sql
INSERT INTO Contractor
--Remaining Code Here
VALUES ('Anderson', 'Vern', GETDATE(), 1)
```



```
  ctrID  LastName  FirstName  HireDate      LocationID
1 5      Anderson  Vern       2012-09-27…   1
                                                                 1 rows
```

Figure 15.25 Insert Vern Anderson into the Contractor table. Use OUTPUT to display the insertion.

Skill Check 3: All six yearly salaried employees are getting a raise of $1500 per year. Run the appropriate UPDATE statement on the PayRates table and show the results on screen. When you are done, your result should resemble Figure 15.26.



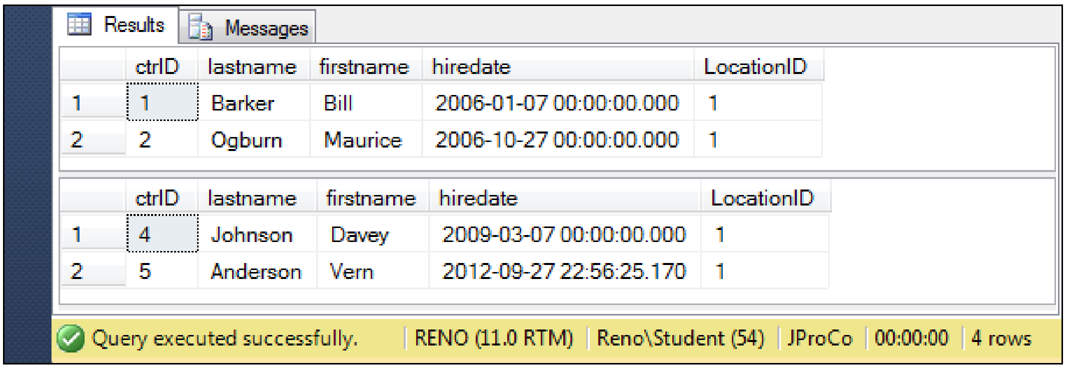
	EmpID	YearlySalary	MonthlySalary	HourlyRate	Selector	Estimate	EmpID	YearlySalary	MonthlySalary	HourlyRate	Selector	Estimate
1	1	99000.00	NULL	NULL	1	1	1	97500.00	NULL	NULL	1	1
2	2	87000.00	NULL	NULL	1	1	2	85500.00	NULL	NULL	1	1
3	6	53500.00	NULL	NULL	1	1	6	52000.00	NULL	NULL	1	1
4	11	116500.00	NULL	NULL	1	1	11	115000.00	NULL	NULL	1	1
5	13	73500.00	NULL	NULL	1	1	13	72000.00	NULL	NULL	1	1
6	14	53500.00	NULL	NULL	NULL	NULL	14	52000.00	NULL	NULL	NULL	NULL

```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (54) | JProCo | 00:00:00 | 6 rows
```

Figure 15.26 UPDATE the PayRates table and use OUTPUT to show the changed records.

Skill Check 4: Due to cutbacks, you need to reduce your contractor workforce. You have decided to keep the two contractors who were hired before January 1, 2007. You will delete any contractor hired after January 1, 2007. You want to send the records affected by this delete operation into a separate table named ContractorLog. Run the code to achieve this and check your Contractor table and your ContractorLog table. Each should have two records, as shown in Figure 15.27.

```sql
SELECT * FROM Contractor
SELECT * FROM ContractorLog
```



	ctrID	lastname	firstname	hiredate	LocationID
1	1	Barker	Bill	2006-01-07 00:00:00.000	1
2	2	Ogburn	Maurice	2006-10-27 00:00:00.000	1

	ctrID	lastname	firstname	hiredate	LocationID
1	4	Johnson	Davey	2009-03-07 00:00:00.000	1
2	5	Anderson	Vern	2012-09-27 22:56:25.170	1

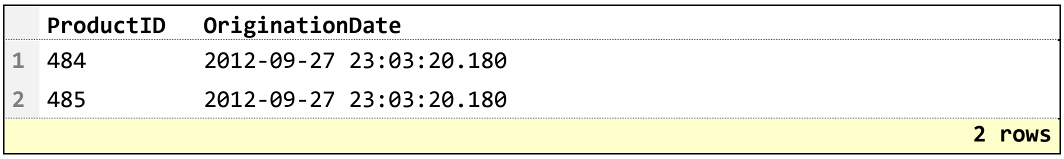
```
Query executed successfully.    RENO (11.0 RTM) | Reno\Student (54) | JProCo | 00:00:00 | 4 rows
```

Figure 15.27 DELETE two records from Contractor. Send the changes to the ContractorLog table.

Skill Check 5: You want to insert two new trips in the CurrentProducts table that have the following insert statement.

```sql
INSERT INTO CurrentProducts VALUES
('Baja 3 Day', 595, GETDATE(), 0, 'Medium-Stay', 0),
('Baja 5 Day', 795, GETDATE(), 0, 'Medium-Stay', 0)
```

You will not need to insert the ID values 484 or 485, because the ProductID in the CurrentProducts table is auto generated by the identity property. You want to use the OUTPUT clause with the INSERT statement to show the two inserted ID's and the OriginationDate in the result. When you are done, your result should resemble Figure 15.28.



```
   ProductID   OriginationDate
1  484         2012-09-27 23:03:20.180
2  485         2012-09-27 23:03:20.180
                                                        2 rows
```

Figure 15.28 Insert two new products and then show the changed records.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab15.1_UsingOutput.sql

Lab 15.2: OUTPUT Code Combinations

Lab Prep: Each lab has one or more Skill Checks. Start with Skill Check 1 and proceed until reaching the Points to Ponder section.

Before beginning this lab, verify that SQL Server 2012 is properly installed and operating. Before running the lab setup script for resetting the database (SQLQueries2012Vol2Chapter15.2Setup.sql), please make sure to close all query windows within SSMS (An open query window pointing to a database context can lock that database preventing it from updating when the script is executing). A simple way to assure all query windows are closed, is to exit out of SSMS, then open a new instance of SSMS, and lastly run the setup script.

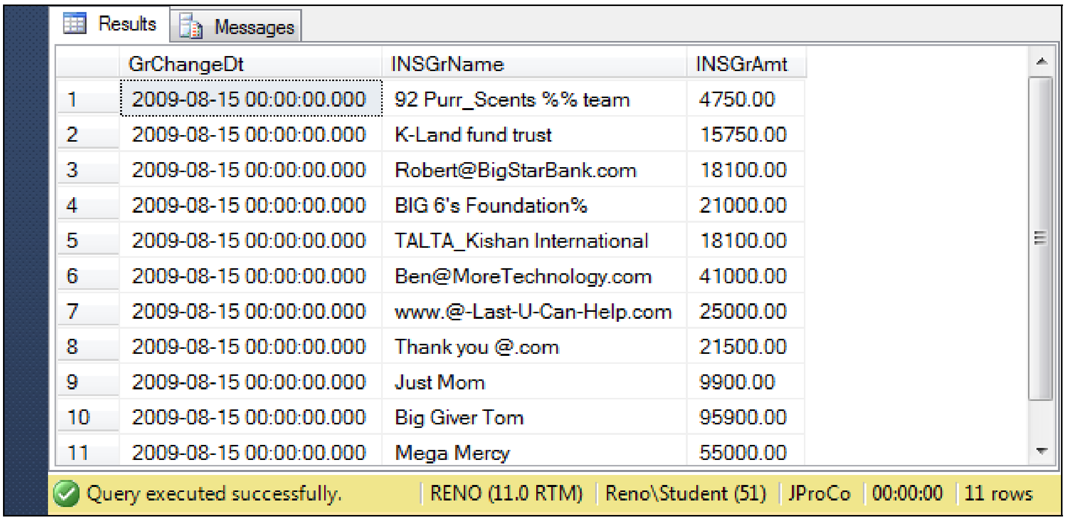
Skill Check 1: Using the MERGE query shown in Figure 15.43, modify the OUTPUT statement to populate into the MyGrantChanges table. Create an archive table called MyGrantChanges to contain your output using the following code:

```
CREATE TABLE MyGrantChanges
(GrChangeDt DATETIME,
INSGrName NVARCHAR(40),
INSGrAmt MONEY)
GO
```

Hard-code the GrChangeDt value to make your result match Figure 15.45. In order to get all 12 records to appear in your output, you will need to reset the Grant table using the sproc (EXEC ResetGrantTables). If you do not reset the Grant table, you will see just 11 records in your result.

If you would like to view the code contained in this sproc, go to **Object Explorer** > **Databases** > **JProCo** > **Programmability** > **Stored Procedures** > right-click **ResetGrantTables** > **Script Stored Procedure as** > **CREATE To** > **New Query Editor Window**.

```
SELECT * FROM MyGrantChanges
```



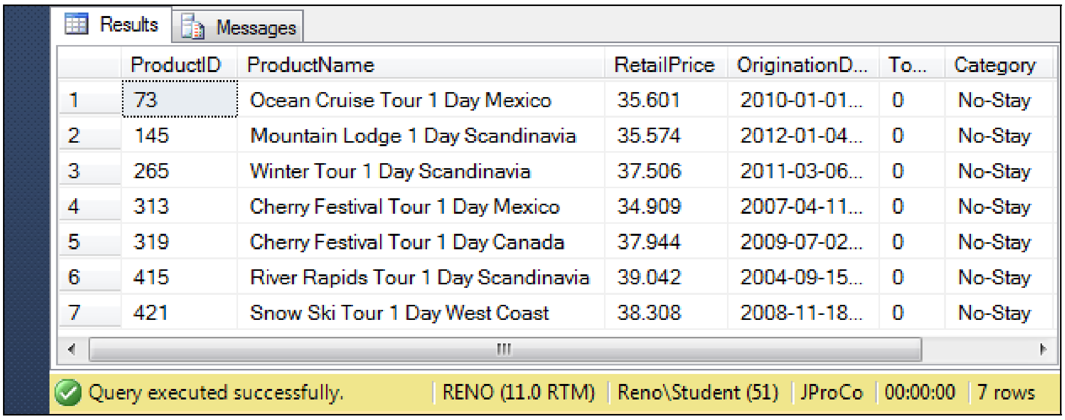
Results | Messages

	GrChangeDt	INSGrName	INSGrAmt
1	2009-08-15 00:00:00.000	92 Purr_Scents %% team	4750.00
2	2009-08-15 00:00:00.000	K-Land fund trust	15750.00
3	2009-08-15 00:00:00.000	Robert@BigStarBank.com	18100.00
4	2009-08-15 00:00:00.000	BIG 6's Foundation%	21000.00
5	2009-08-15 00:00:00.000	TALTA_Kishan International	18100.00
6	2009-08-15 00:00:00.000	Ben@MoreTechnology.com	41000.00
7	2009-08-15 00:00:00.000	www.@-Last-U-Can-Help.com	25000.00
8	2009-08-15 00:00:00.000	Thank you @.com	21500.00
9	2009-08-15 00:00:00.000	Just Mom	9900.00
10	2009-08-15 00:00:00.000	Big Giver Tom	95900.00
11	2009-08-15 00:00:00.000	Mega Mercy	55000.00

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 11 rows

Figure 15.44 Modify the output from 15.43 and send it to the storage table MyGrantChanges.

Skill Check 2: You have seven products that each cost less than $40.



Results | Messages

	ProductID	ProductName	RetailPrice	OriginationD...	To...	Category
1	73	Ocean Cruise Tour 1 Day Mexico	35.601	2010-01-01...	0	No-Stay
2	145	Mountain Lodge 1 Day Scandinavia	35.574	2012-01-04...	0	No-Stay
3	265	Winter Tour 1 Day Scandinavia	37.506	2011-03-06...	0	No-Stay
4	313	Cherry Festival Tour 1 Day Mexico	34.909	2007-04-11...	0	No-Stay
5	319	Cherry Festival Tour 1 Day Canada	37.944	2009-07-02...	0	No-Stay
6	415	River Rapids Tour 1 Day Scandinavia	39.042	2004-09-15...	0	No-Stay
7	421	Snow Ski Tour 1 Day West Coast	38.308	2008-11-18...	0	No-Stay

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 7 rows

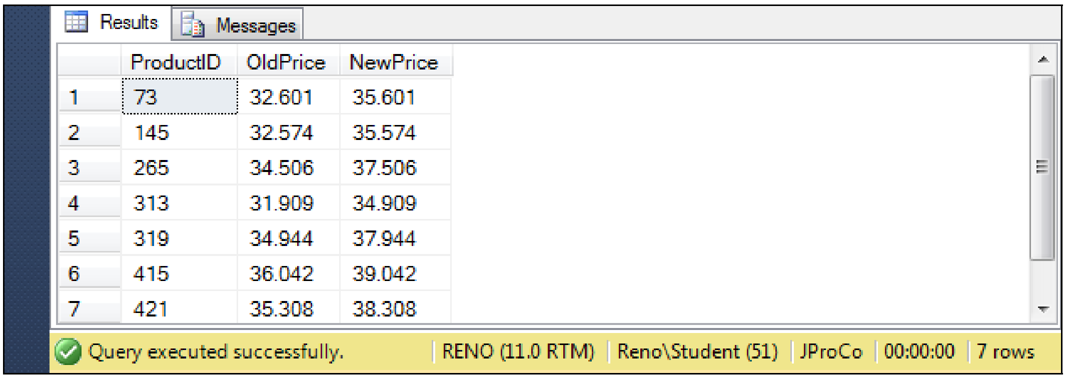
Figure 15.45 Seven products in CurrentProducts currently cost less than $40.

You plan to write the following update statement to raise the price of each of these products by $3.00:

```
UPDATE CurrentProducts
SET RetailPrice = RetailPrice + 3.00
WHERE RetailPrice < 40
```

You want to capture the results of the update into a table called ProductPriceChange. Capture the ProductID and OldPrice from the Deleted table and NewPrice from the Inserted table. When you are done, you should query the ProductPriceChange table and see the results as seen in Figure 15.45. Hint: We can accomplish a correct solution without a derived table and an INSERT INTO statement, but it is highly recommended that you also attempt it using the method demonstrated in this chapter.

```
SELECT * FROM ProductPriceChange
```



Results | Messages

	ProductID	OldPrice	NewPrice
1	73	32.601	35.601
2	145	32.574	35.574
3	265	34.506	37.506
4	313	31.909	34.909
5	319	34.944	37.944
6	415	36.042	39.042
7	421	35.308	38.308

Query executed successfully. | RENO (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 7 rows

Figure 15.46 This shows OldPrice and NewPrice.

Skill Check 3: When you need to test some code but don't want to experiment on a live table, this code quickly creates a copy of a table. Be sure to run this code before completing the Skill Check:

```
SELECT * INTO ImportantTableCopy
FROM ImportantTable
```

Use a similar query to create a test copy of the Employee table:

```
SELECT * INTO EmployeeLMO
FROM Employee
```

Use MERGE to update EmployeeLMO (alias as ELMO) with the changes contained in EmpCheckMaster (alias as ECM). ECM is a reconciliation table prepared at the start of each fiscal year to update the master employee list once performance reviews and promotions have been processed. Your MERGE query should ensure that all changes contained in ECM are carried over to ELMO. In fact, when the MERGE is complete, ELMO and ECM should be identical.

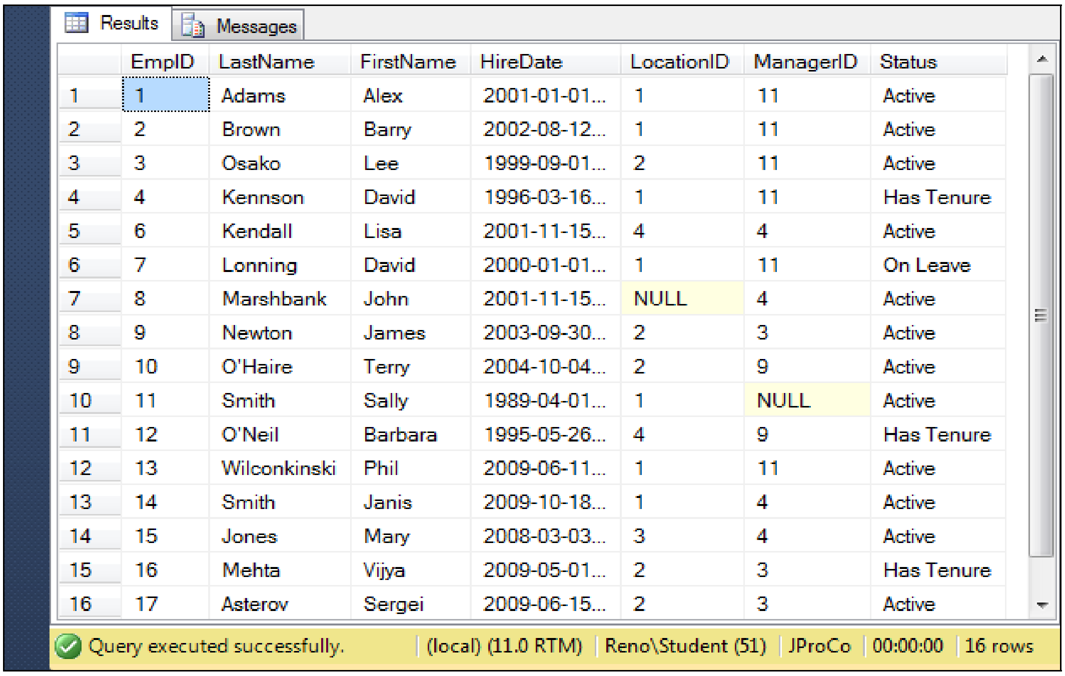
Insert any records in the source table (EmpCheckMaster) not found in the target (ELMO). Update any common records between the two tables, so that matched records in ELMO will become identical to those in ECM. If there are any records in the target not contained in the source, delete them. Before it processes any changes to matched records, your code must first confirm there has been a change in ManagerID. Any matched records not containing a manager change should be ignored and not affected by the MERGE.

Use OUTPUT to contain the records inserted during the MERGE. Make sure your OUTPUT result populates the archive table EmpMergeArchive and the fields appear in the order shown in Figure 15.47. Be sure to run the queries at the beginning of this Skill Check or you will get the following error message.

```
Messages
Msg 208, Level 16, State 1, Line 1
Invalid object name 'EmpCheckMaster'
                                                          0 rows
```

Figure 15.47 If you see this error message run the query at the beginning of Skill Check 3.

```sql
SELECT * FROM EmployeeLMO
```



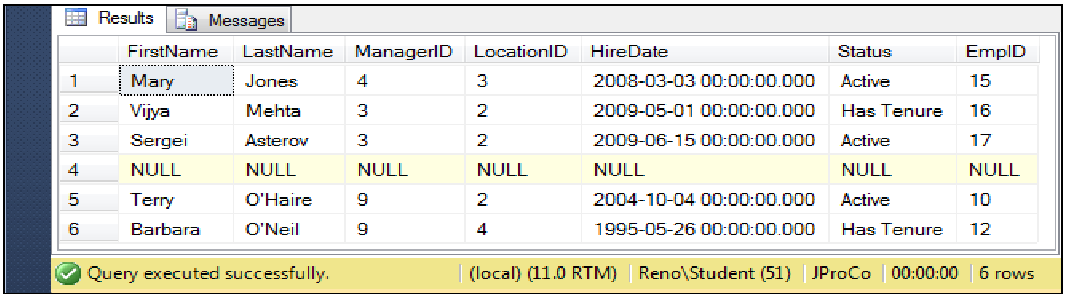
Results | Messages

	EmpID	LastName	FirstName	HireDate	LocationID	ManagerID	Status
1	1	Adams	Alex	2001-01-01...	1	11	Active
2	2	Brown	Barry	2002-08-12...	1	11	Active
3	3	Osako	Lee	1999-09-01...	2	11	Active
4	4	Kennson	David	1996-03-16...	1	11	Has Tenure
5	6	Kendall	Lisa	2001-11-15...	4	4	Active
6	7	Lonning	David	2000-01-01...	1	11	On Leave
7	8	Marshbank	John	2001-11-15...	NULL	4	Active
8	9	Newton	James	2003-09-30...	2	3	Active
9	10	O'Haire	Terry	2004-10-04...	2	9	Active
10	11	Smith	Sally	1989-04-01...	1	NULL	Active
11	12	O'Neil	Barbara	1995-05-26...	4	9	Has Tenure
12	13	Wilconkinski	Phil	2009-06-11...	1	11	Active
13	14	Smith	Janis	2009-10-18...	1	4	Active
14	15	Jones	Mary	2008-03-03...	3	4	Active
15	16	Mehta	Vijya	2009-05-01...	2	3	Has Tenure
16	17	Asterov	Sergei	2009-06-15...	2	3	Active

Query executed successfully. | (local) (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 16 rows

Figure 15.48 EmployeeLMO is now a table in JProCo.

```sql
SELECT * FROM EmpMergeArchive
```



Results | Messages

	FirstName	LastName	ManagerID	LocationID	HireDate	Status	EmpID
1	Mary	Jones	4	3	2008-03-03 00:00:00.000	Active	15
2	Vijya	Mehta	3	2	2009-05-01 00:00:00.000	Has Tenure	16
3	Sergei	Asterov	3	2	2009-06-15 00:00:00.000	Active	17
4	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	Terry	O'Haire	9	2	2004-10-04 00:00:00.000	Active	10
6	Barbara	O'Neil	9	4	1995-05-26 00:00:00.000	Has Tenure	12

Query executed successfully. | (local) (11.0 RTM) | Reno\Student (51) | JProCo | 00:00:00 | 6 rows

Figure 15.49 Use OUTPUT to send the results of your MERGE into EmpMergeArchive.

Answer Code: The T-SQL code for this lab is located in the downloadable files as a file named Lab15.2_OutputCodeCombinations.sql