The Labs are from the Book

To Do The Labs: Follow the progression of explanatory material in the book. When you reach a code section (example or demonstration) copy the section from here into the query editor window.

(Note: Sometimes you may have Create Table and INSERT statements to run before the query.)

You could run your code now. However, a far more effective technique would be to now type the query again directly under the copied query (what you are typing is what is directly in front of you on the screen.) Then, run what you typed. This helps a lot with retaining the skill/knowledge!

Practice, Practice, and More Practice

The only way to really learn T-SQL is by doing it.

You will not learn T-SQL without practicing it yourself.

The more you practice and experiment, the faster you will learn.

Complete these labs:

Type and Execute each query in SSMS.

These Labs are taken directly from the text, Beginning T-SQL 2012.

The Labs are presented here for your convenience, but they are the same as found throughout the current reading chapter in the book.

You will be best served if you follow along in the current chapter and complete the examples provided along with working to understand the explanation of how it works.

This technique (Learning By Example) has been found to be highly effective when used as described in this guide.

As much as possible, type each query.

Manually typing in each query will help you to develop a solid sense of the syntax and some muscle memory (best for retention) and is a much more effective learning strategy than cut and paste will be!
Assignment Grading Rubric

Course: IT350 Unit: 8    Points: 15

Note ** Using the split screen feature in Windows 8 can be a great time saving strategy.

Also, for all your online education work, an optimal setup will be to have two separate monitors so that you can open and see the book, the course (Lab), and SSMS all at the same time. This is a low cost solution that can pay big dividends in your productivity, time, and learning effectiveness!

Provide a screenshot from SSMS showing the queries for each main section below.

--Create Demo Table

-- Note -- If you do not provide a schema name such as "Sales" (Sales.DemoCustomer) SQL Server automatically assigns the "dbo" schema
-- You may need to right click on tables and select refresh in order to see your new tables.

USE AdventureWorks2012;
GO
IF OBJECT_ID('demoCustomer') IS NOT NULL BEGIN
   DROP TABLE demoCustomer;
END;
CREATE TABLE demoCustomer(CustomerID INT NOT NULL PRIMARY KEY, FirstName NVARCHAR(50) NOT NULL, MiddleName NVARCHAR(50) NOT NULL, LastName NVARCHAR(50) NOT NULL);

-- 1. Create a table called dbo.testProvider. Include a ProviderID that is an identity column and is a primary key. Include Company name, PrimaryContactName and an AuthorizedDate column with a check constraint specifying that the value must be > Dec 12, 1999 and not greater than current date. Also Include a column called "Active" that is one character with a default of Y and allows only Y or N.

--Adding One Row at a Time with Literal Values

USE AdventureWorks2012;
GO
--1
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName) VALUES (1,'Orlando','N.','Gee');
--2
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName) SELECT 3,'Donna','F.','Cameras';
--3
INSERT INTO dbo.demoCustomer VALUES (4,'Janet','M.','Gates');
--4
INSERT INTO dbo.demoCustomer SELECT 6,'Rosmarie','J.','Carroll';
--5
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName) VALUES (2,'Keith',NULL,'Harris');
--6
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, LastName) VALUES (5,'Lucy','Harrington');
SELECT CustomerID, FirstName, MiddleName, LastName
FROM dbo.demoCustomer;

-- 2. Insert three rows (individually) into the testProvider Table you created above. Remember that
providerID is an Identity column.

-- Attempting to Insert Rows with Invalid INSERT Statements-- These are examples of how to do it wrong.
USE AdventureWorks2012;
GO
PRINT '1';

--1
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
VALUES (1, 'Dominic','P.','Gash');
PRINT '2';
--2
INSERT INTO dbo.demoCustomer (CustomerID, MiddleName, LastName)
VALUES (10,'M.','Garza');
GO
PRINT '3';
GO
--3
INSERT INTO dbo.demoCustomer
VALUES (11,'Katherine','Harding');
GO
PRINT '4';
GO
--4
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, LastName)
VALUES (11, 'Katherine', NULL,'Harding');
GO
PRINT '5';
GO
--5
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, LastName)
VALUES ('A','Katherine','Harding');

-- Inserting Multiple Rows with One INSERT
USE AdventureWorks2012;
GO
-- hint -- You will recognize that the Union operator creates a dataset using rows (from multiple
tables or, as in this case, from row literals)
-- rather than by creating a result set using JOIN that selects columns rather than rows.
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
SELECT 7,'Dominic','P.','Gash'
UNION
SELECT 10,'Kathleen','M.','Garza'
UNION
SELECT 11, 'Katherine', NULL,'Harding';

--2
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
VALUES (12,'Johnny','A.','Capino'),
(16, 'Christopher', 'R.', 'Beck'),
(18, 'David', 'J.', 'Liu');

--3
SELECT CustomerID, FirstName, MiddleName, LastName
FROM dbo.demoCustomer
WHERE CustomerID >= 7;

-- 3. Insert four rows into the dbo.testProvider table using only one insert statement,
-- Make sure that the middle name column value is null for all four rows.

--Inserting Rows from Another Table

USE AdventureWorks2012;
GO
--1
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
SELECT BusinessEntityID, FirstName, MiddleName, LastName
FROM Person.Person
WHERE BusinessEntityID BETWEEN 19 AND 35;

--2
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
SELECT DISTINCT c.BusinessEntityID, c.FirstName, c.MiddleName, c.LastName
FROM Person.Person AS c
INNER JOIN Sales.SalesOrderHeader AS s ON c.BusinessEntityID = s.SalesPersonID

--3
SELECT CustomerID, FirstName, MiddleName, LastName
FROM dbo.demoCustomer
WHERE CustomerID > 18;

-- 4. You want to populate the testProvider table with some of the rows from the dbo.demoCustomer
-- table,
-- Create a query to accomplish this, only insert rows that have a "Null" middleName value.

--Inserting Missing Rows

--USE AdventureWorks2012;
--GO
-----1
--INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
--SELECT c.BusinessEntityID, c.FirstName, c.MiddleName, c.LastName
--FROM Person.Person AS c
--LEFT OUTER JOIN dbo.demoCustomer AS d ON c.BusinessEntityID = d.CustomerID
--WHERE d.CustomerID IS NULL;

--2
SELECT COUNT(CustomerID) AS CustomerCount
FROM dbo.demoCustomer
WHERE CustomerID > 18;

-- 5. Describe your detailed understanding of what, exactly and specifically, happens in the above query
-- Use a step by step approach starting with the inner select statement.

--Using SELECT INTO to Create and Populate a Table

USE AdventureWorks2012;

Copyright Kaplan University
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoCustomer]')
AND type in (N'U'))
DROP TABLE dbo.demoCustomer;
GO

--1
--SELECT BusinessEntityID, FirstName, MiddleName, LastName,
--FirstName + ISNULL(' ' + MiddleName,'') + ' ' + LastName AS FullName
--INTO dbo.demoCustomer
--FROM Person.Person;
--2
SELECT BusinessEntityID, FirstName, MiddleName, LastName, FullName
FROM dbo.demoCustomer;

--6. What is the purpose and results of using ISNULL in the above statement?

--Inserting Data with a Column Default Constraint

USE AdventureWorks2012;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoDefault]')
AND type in (N'U'))
DROP TABLE [dbo].[demoDefault]
GO
CREATE TABLE [dbo].[demoDefault](
[KeyColumn] [int] NOT NULL PRIMARY KEY,
[HasADefault1] [DATETIME2](1) NOT NULL,
[HasADefault2] [NVARCHAR](50) NULL,
)
GO
ALTER TABLE [dbo].[demoDefault] ADD CONSTRAINT [DF_demoDefault_HasADefault1]
DEFAULT (GETDATE()) FOR [HasADefault1]
GO
ALTER TABLE [dbo].[demoDefault] ADD CONSTRAINT [DF_demoDefault_HasADefault2]
DEFAULT ('the default') FOR [HasADefault2]
GO
--1
INSERT INTO dbo.demoDefault(HasADefault1,HasADefault2,KeyColumn)
VALUES ('2009-04-24','Test 1',1),('2009-10-1',NULL,2);
--2
INSERT INTO dbo.demoDefault (HasADefault1,HasADefault2,KeyColumn)
VALUES (DEFAULT,DEFAULT,3),(DEFAULT,DEFAULT,4);
--3
INSERT INTO dbo.demoDefault (KeyColumn)
VALUES (5),(6);
--4
SELECT HasADefault1,HasADefault2,KeyColumn
FROM dbo.demoDefault;

--7. Insert the following values into the demoDefault Table you created in the lab. --- Row 1
HasADefault1 = NULL, HasADefault2 = "George", KeyColumn (8),          ---Row 2 HasADefault1 = ' ',
KeyColumn (9)
--- Query the resulting records. Contrast the results obtained.

--Inserting Rows into Tables with Auto populated Columns

USE [AdventureWorks2012]
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'N'[dbo].[demoAutoPopulate]')
AND type in (N'U'))
DROP TABLE [dbo].[demoAutoPopulate];
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'N'[dbo].[demoSequence]'))
DROP SEQUENCE [dbo].[demoSequence];
CREATE SEQUENCE dbo.demoSequence
AS INT
START WITH 1
INCREMENT BY 1;
CREATE TABLE [dbo].[demoAutoPopulate](
[RegularColumn] [NVARCHAR](50) NOT NULL PRIMARY KEY,
[IdentityColumn] [INT] IDENTITY(1,1) NOT NULL,
[RowversionColumn] [ROWVERSION] NOT NULL,
[SequenceColumn] [INT] NOT NULL,
[ComputedColumn] AS ([RegularColumn]+CONVERT([NVARCHAR],
[IdentityColumn],(1))) PERSISTED)
GO
--1
INSERT INTO dbo.demoAutoPopulate (RegularColumn, SequenceColumn)
VALUES ('a', NEXT VALUE FOR dbo.demoSequence),
('b', NEXT VALUE FOR dbo.demoSequence),
('c', NEXT VALUE FOR dbo.demoSequence);
--2
SELECT RegularColumn, IdentityColumn, RowversionColumn, SequenceColumn, ComputedColumn
FROM demoAutoPopulate;

-- 8 Alter the demoSequence table from above so that the sequence starts at 10 and increases by 1.
Rerun the code above and display your results
-- Explain why there are only 2 values being inserted into the table, and yet the table shows 5 values.

USE AdventureWorks2012;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'N'[dbo].[demoProduct]')
AND type in (N'U'))
DROP TABLE [dbo].[demoProduct]
GO
CREATE TABLE [dbo].[demoProduct](
[ProductID] [INT] NOT NULL PRIMARY KEY,
[Name] [dbo].[Name] NOT NULL,
[Color] [NVARCHAR](15) NULL,
[StandardCost] [MONEY] NOT NULL,
[ListPrice] [MONEY] NOT NULL,
[Size] [NVARCHAR](5) NULL,
[Weight] [DECIMAL](8, 2) NULL,
);
IF EXISTS (SELECT * FROM sys.objects
Copyright Kaplan University
WHERE object_id = OBJECT_ID(N'[dbo].[demoSalesOrderHeader]')
AND type in (N'U'))
DROP TABLE [dbo].[demoSalesOrderHeader]
GO
CREATE TABLE [dbo].[demoSalesOrderHeader](
    [SalesOrderID] [INT] NOT NULL PRIMARY KEY,
    [SalesID] [INT] NOT NULL IDENTITY,
    [OrderDate] [DATETIME] NOT NULL,
    [CustomerID] [INT] NOT NULL,
    [SubTotal] [MONEY] NOT NULL,
    [TaxAmt] [MONEY] NOT NULL,
    [Freight] [MONEY] NOT NULL,
    [DateEntered] [DATETIME],
    [SalesNumber] [INT] NOT NULL,
    [TotalDue] AS (ISNULL(([SubTotal]+[TaxAmt])+[Freight],(0))),
    [RV] ROWVERSION NOT NULL);
GO
ALTER TABLE [dbo].[demoSalesOrderHeader] ADD CONSTRAINT
[DF_demoSalesOrderHeader_DateEntered] DEFAULT (getdate()) FOR [DateEntered];
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoSalesSequence]'))
DROP SEQUENCE [dbo].[demoSalesSequence]
GO
CREATE SEQUENCE demoSalesSequence
AS INT
START WITH 1
INCREMENT BY 1;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoAddress]')
AND type in (N'U'))
DROP TABLE [dbo].[demoAddress]
GO
CREATE TABLE [dbo].[demoAddress](
    [AddressID] [INT] NOT NULL IDENTITY PRIMARY KEY,
    [AddressLine1] [NVARCHAR](60) NOT NULL,
    [AddressLine2] [NVARCHAR](60) NULL,
    [City] [NVARCHAR](30) NOT NULL,
    [PostalCode] [NVARCHAR](15) NOT NULL
);
---Creating Demo Tables
USE AdventureWorks2012;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoProduct]')
AND type in (N'U'))
DROP TABLE [dbo].[demoProduct]
GO
SELECT * INTO dbo.demoProduct FROM Production.Product;
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoCustomer]')
AND type in (N'U'))
Copyright Kaplan University
DROP TABLE [dbo].[demoCustomer];
GO
SELECT * INTO dbo.demoCustomer FROM Sales.Customer;
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoAddress]')
AND type in (N'U'))
DROP TABLE [dbo].[demoAddress];
GO
SELECT * INTO dbo.demoAddress FROM Person.Address;
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoSalesOrderHeader]')
AND type in (N'U'))
DROP TABLE [dbo].[demoSalesOrderHeader];
GO
SELECT * INTO dbo.demoSalesOrderHeader FROM Sales.SalesOrderHeader;
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoSalesOrderDetail]')
AND type in (N'U'))
DROP TABLE [dbo].[demoSalesOrderDetail];
GO
SELECT * INTO dbo.demoSalesOrderDetail FROM Sales.SalesOrderDetail;

USE AdventureWorks2012;
SELECT * INTO Sales.SalesOrderDetail FROM dbo.demoSalesOrderDetail;

USE AdventureWorks2012;
GO
--1
SELECT CustomerID
FROM dbo.demoCustomer;
--2
DELETE dbo.demoCustomer;
--3
SELECT CustomerID
FROM dbo.demoCustomer;
--4
SELECT ProductID
FROM dbo.demoProduct
WHERE ProductID > 900;
--5
DELETE dbo.demoProduct
WHERE ProductID > 900;
--6
SELECT ProductID
FROM dbo.demoProduct
WHERE ProductID > 900;

--Deleting When Joining or Using a Subquery
USE AdventureWorks2012;
GO
--1
SELECT d.SalesOrderID, SalesOrderNumber
Copyright Kaplan University
FROM dbo.demoSalesOrderDetail AS d
INNER JOIN dbo.demoSalesOrderHeader AS h ON d.SalesOrderID = h.SalesOrderID
WHERE h.SalesOrderNumber = 'SO71797'
--2
DELETE d
FROM dbo.demoSalesOrderDetail AS d
INNER JOIN dbo.demoSalesOrderHeader AS h ON d.SalesOrderID = h.SalesOrderID
WHERE h.SalesOrderNumber = 'SO71797'
--3
SELECT d.SalesOrderID, SalesOrderNumber
FROM dbo.demoSalesOrderDetail AS d
INNER JOIN dbo.demoSalesOrderHeader AS h ON d.SalesOrderID = h.SalesOrderID
WHERE h.SalesOrderNumber = 'SO71797'
--4
SELECT SalesOrderID, ProductID
FROM dbo.demoSalesOrderDetail
WHERE ProductID NOT IN
(SELECT ProductID FROM dbo.demoProduct WHERE ProductID IS NOT NULL);
--5
DELETE FROM dbo.demoSalesOrderDetail
WHERE ProductID NOT IN
(SELECT ProductID FROM dbo.demoProduct WHERE ProductID IS NOT NULL);
--6
SELECT SalesOrderID, ProductID
FROM dbo.demoSalesOrderDetail
WHERE ProductID NOT IN
(SELECT ProductID FROM dbo.demoProduct WHERE ProductID IS NOT NULL);
--7

---Truncating Tables
USE AdventureWorks2012;
GO
--1
SELECT SalesOrderID, OrderDate
FROM dbo.demoSalesOrderHeader;
--2
TRUNCATE TABLE dbo.demoSalesOrderHeader;
--3
SELECT SalesOrderID, OrderDate
FROM dbo.demoSalesOrderHeader;

---Updating Data in a Table
USE AdventureWorks2012;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoPerson]')
AND type in (N'U'))
DROP TABLE [dbo].[demoPerson]
GO
SELECT * INTO dbo.demoPerson
FROM Person.Person
WHERE Title in ('Mr.', 'Mrs.', 'Ms.');
Assignment Grading Rubric
Course: IT350 Unit: 8   Points: 15

```sql
SELECT BusinessEntityID, NameStyle, Title
FROM dbo.demoPerson
ORDER BY BusinessEntityID;
--2
UPDATE dbo.demoPerson
SET NameStyle = 1;
--3
SELECT BusinessEntityID, NameStyle, Title
FROM dbo.demoPerson
ORDER BY BusinessEntityID;
--4
UPDATE dbo.demoPerson
SET NameStyle = 0
WHERE Title = 'Ms.';
--5
SELECT BusinessEntityID, NameStyle, Title
FROM dbo.demoPerson
ORDER BY BusinessEntityID;

--Update with Expressions, Columns, or Data from Another Table

USE AdventureWorks2012;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoPersonStore]')
AND type in (N'U'))
DROP TABLE [dbo].[demoPersonStore]
GO
CREATE TABLE [dbo].[demoPersonStore] (
    [FirstName] [NVARCHAR] (60),
    [LastName] [NVARCHAR] (60),
    [CompanyName] [NVARCHAR] (60)
);
INSERT INTO dbo.demoPersonStore (FirstName, LastName, CompanyName)
SELECT a.FirstName, a.LastName, c.Name
FROM Person.Person a
JOIN Sales.SalesPerson b
ON a.BusinessEntityID = b.BusinessEntityID
JOIN Sales.Store c
ON b.BusinessEntityID = c.SalesPersonID
--1
SELECT FirstName, LastName, CompanyName,
LEFT(FirstName,3) + '.' + LEFT(LastName,3) AS NewCompany
FROM dbo.demoPersonStore;
--2
UPDATE dbo.demoPersonStore
SET CompanyName = LEFT(FirstName,3) + '.' + LEFT(LastName,3);
--3
SELECT FirstName, LastName, CompanyName,
LEFT(FirstName,3) + '.' + LEFT(LastName,3) AS NewCompany
FROM dbo.demoPersonStore;

Updating with a Join
Copyright Kaplan University
USE AdventureWorks2012;
GO
--1
SELECT AddressLine1, AddressLine2
FROM dbo.demoAddress;
--2
UPDATE a
SET AddressLine1 = FirstName + ' ' + LastName,
AddressLine2 = AddressLine1 + ISNULL(' ' + AddressLine2,'')
FROM dbo.demoAddress AS a
INNER JOIN Person.BusinessEntityAddress c ON a.AddressID = c.AddressID
INNER JOIN Person.Person b ON b.BusinessentityID = c.BusinessEntityID
--3
SELECT AddressLine1, AddressLine2
FROM dbo.demoAddress;

Updates with Aggregate Expressions

USE AdventureWorks2012;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoCustomerSummary]')
AND type in (N'U'))
DROP TABLE [dbo].[demoCustomerSummary];
GO
CREATE TABLE dbo.demoCustomerSummary (CustomerID INT NOT NULL PRIMARY KEY,
SaleCount INTEGER NULL,
TotalAmount MONEY NULL);
GO
INSERT INTO dbo.demoCustomerSummary (CustomerID, SaleCount,TotalAmount)
SELECT BusinessEntityID, 0, 0
FROM dbo.demoPerson;
GO
--1
SELECT CustomerID, SaleCount, TotalAmount
FROM dbo.demoCustomerSummary
WHERE CustomerID in (11621,12798,13589,14465,18623);
--2
WITH Totals AS (
SELECT COUNT(*) AS SaleCount,SUM(TotalDue) AS TotalAmount,
CustomerID
FROM dbo.demoSalesOrderHeader
GROUP BY CustomerID)
UPDATE c SET TotalAmount = Totals.TotalAmount,
SaleCount = Totals.SaleCount
FROM dbo.demoCustomerSummary AS c
INNER JOIN Totals ON c.CustomerID = Totals.CustomerID;
--3
SELECT CustomerID, SaleCount, TotalAmount
FROM dbo.demoCustomerSummary
WHERE CustomerID in (11621,12798,13589,14465,18623);
Explicit Transactions

USE AdventureWorks2012;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoTransaction]')
AND type in (N'U'))
DROP TABLE [dbo].[demoTransaction];
GO
CREATE TABLE dbo.demoTransaction (col1 INT NOT NULL);
GO
--1
BEGIN TRAN
INSERT INTO dbo.demoTransaction (col1) VALUES (1);
INSERT INTO dbo.demoTransaction (col1) VALUES (2);
COMMIT TRAN
--2
BEGIN TRAN
INSERT INTO dbo.demoTransaction (col1) VALUES (3);
INSERT INTO dbo.demoTransaction (col1) VALUES ('a');
COMMIT TRAN
GO
--3
SELECT col1
FROM dbo.demoTransaction;

USE AdventureWorks2012;
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoTransaction]')
AND type in (N'U'))
DROP TABLE [dbo].[demoTransaction];
GO
CREATE TABLE dbo.demoTransaction (col1 INT NOT NULL);
GO
--1
BEGIN TRAN
INSERT INTO dbo.demoTransaction (col1) VALUES (1);
INSERT INTO dbo.demoTransaction (col1) VALUES (2);
COMMIT TRAN
--2
BEGIN TRAN
INSERT INTO dbo.demoTransaction (col1) VALUES (3);
INSERT INTO dbo.demoTransaction (col1) VALUES (4);
ROLLBACK TRAN
GO
--3
SELECT col1
FROM dbo.demoTransaction;

The Difference Between the Set-Based and Iterative Approaches

USE AdventureWorks2012;
Copyright Kaplan University
GO
--Create a work table
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoPerformance]')
AND type in (N'U'))
DROP TABLE [dbo].[demoPerformance];
GO
CREATE TABLE [dbo].[demoPerformance](
    [SalesOrderID] [int] NOT NULL,
    [SalesOrderDetailID] [int] NOT NULL,
    CONSTRAINT [PK_demoPerformance] PRIMARY KEY CLUSTERED
    ( [SalesOrderID] ASC,
      [SalesOrderDetailID] ASC
    ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
      ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
PRINT 'Insert all rows start';
PRINT getdate();
--Insert all rows from the Sales.SalesOrderDetail table at once
INSERT INTO dbo.demoPerformance(SalesOrderID, SalesOrderDetailID)
SELECT SalesOrderID, SalesOrderDetailID
FROM Sales.SalesOrderDetail;
PRINT 'Insert all rows end';
PRINT getdate();
--Remove all rows from the first insert
TRUNCATE TABLE [dbo].[demoPerformance];
PRINT 'Insert rows one at a time begin';
PRINT getdate();
--Set up a loop to insert one row at a time
WHILE EXISTS(
    SELECT *
    FROM Sales.SalesOrderDetail AS d LEFT JOIN dbo.demoPerformance AS p
    ON d.SalesOrderID = p.SalesOrderID
    AND d.SalesOrderDetailID = p.SalesOrderDetailID
    WHERE p.SalesOrderID IS NULL)
BEGIN
    INSERT INTO dbo.demoPerformance (SalesOrderID,SalesOrderDetailID)
    SELECT TOP 1 d.SalesOrderID, d.SalesOrderDetailID
    FROM Sales.SalesOrderDetail AS d LEFT JOIN dbo.demoPerformance AS p
    ON d.SalesOrderID = p.SalesOrderID
    AND d.SalesOrderDetailID = p.SalesOrderDetailID
    WHERE p.SalesOrderID IS NULL;
END
PRINT 'Insert rows one at a time end';
PRINT getdate();

Listing 6-20. Deleting Demo Tables
USE [AdventureWorks2012];
GO
IF EXISTS (SELECT * FROM sys.objects
WHERE object_id = OBJECT_ID(N'[dbo].[demoProduct]')
AND type in (N'U'))
DROP TABLE [dbo].[demoProduct];
Submit to the appropriate Dropbox.

You will earn 15 pts. for submitting your best attempt.

Copyright Kaplan University
The labs are intended to help you practice and think about the concepts in this course.

You will receive full grades for submitting your attempted answers.

Remember to type them in and avoid cut and paste where possible / practical.

**Directions:**

- When you have completed the Reading:
  - Open a word document.
  - Copy / Paste the section Descriptor.
  - Paste a screenshot (snipping tool) of the code executed in SSMS.

Your professor will review your Lab work in order to determine how you are progressing in the course. He/she may or may not provide specific feedback. In addition, the professor will use the Labs to determine if there are areas of covered material that many students are having difficulty with, and may then decide to provide extra explanation in the next Seminar or discussion. Review the grading rubric below before beginning this Activity.

**Assignment grading rubric = 15 points**

<table>
<thead>
<tr>
<th>Assignment Criteria</th>
<th>Points possible</th>
<th>Points earned by student</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Includes a student response to all practice questions code samples / demonstrations showing that the code was run in SSMS questions.</td>
<td>0 - 15</td>
<td></td>
</tr>
</tbody>
</table>

**Column Total**

Copyright Kaplan University