Cis 111 Chapter 6 Student Lecture Notes

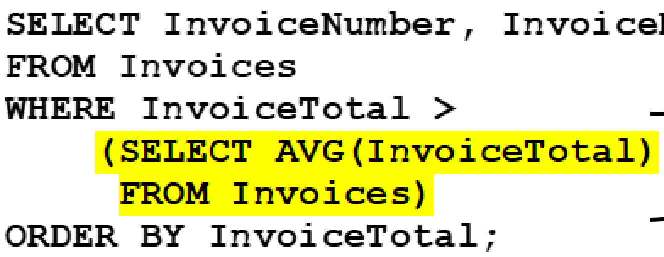
Topic #1: Subquery – Where Clause

A **subquery** is a Select statement introduced (coded) within another Select statement. There are four ways to code a subquery in a Select statement:

- In a **Where clause** as a search condition.
- In a **Having clause** as a search condition.
- In a **From clause** as a table specification.
- In the **Select clause** as a column specification.

Where clause subquery

Example #1



```
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal
FROM Invoices
WHERE InvoiceTotal >
    (SELECT AVG(InvoiceTotal)
     FROM Invoices)
ORDER BY InvoiceTotal;
```

The subquery coded within the Where clause will be executed first. What will this subquery return? (choose one)

- □ a single value
- □ a column of values
- □ a table

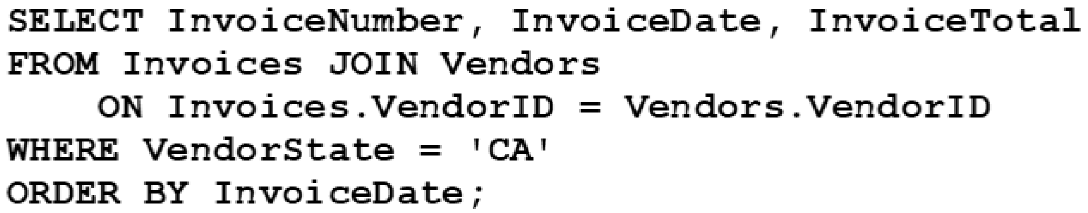
The first Select statement is referred to as an **outer query,** which means, it will be executed after the subquery is complete. The Where clause for the outer query contains the syntax **Where InvoiceTotal > (subquery).** This is the reason the subquery had to return a single value. If the subquery returns a column of values or a table then a SQL exception (error) would occur.

Lab Activity

1. Launch SSMS and connect to the Cis111 AP database.
2. Click New Query button.
    a. Make comment line and label it Chapter 6 Exercises.
    b. Make another comment line and label it –Query #2
3. Create a subquery for exercise #2 on page 212.

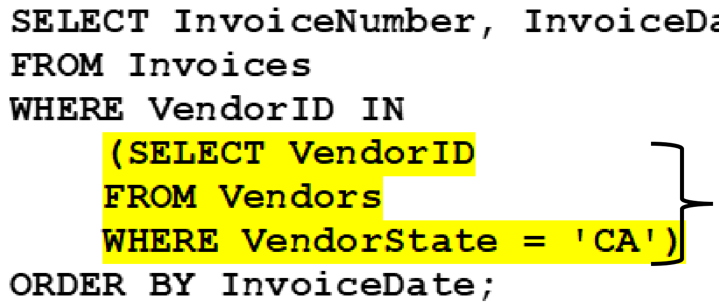
Example #2

For the next example you will first review a Select statement that uses an inner join to display invoice information for vendors located in the state of California. It does not use a subquery:



```
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal
FROM Invoices JOIN Vendors
    ON Invoices.VendorID = Vendors.VendorID
WHERE VendorState = 'CA'
ORDER BY InvoiceDate;
```

You will now review a subquery that returns the same information as the above query without using an inner join:



```
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal
FROM Invoices
WHERE VendorID IN
    (SELECT VendorID
    FROM Vendors
    WHERE VendorState = 'CA')
ORDER BY InvoiceDate;
```

The subquery coded within the Where clause will return all vendor ids located in the state of California. What will this subquery return? (choose one)

- □ a single value
- □ a column of values
- □ a table

The Where clause for the outer query contains the syntax **Where VendorID In (subquery).** This allows the subquery to return one or more values, or a column of values, for a single field.

Lab Activity

4. Create a subquery for exercise #1 on page 212.
5. What adjustment would be made to list vendor names who have not submitted any invoice?
    a. What type of join could be used for this problem to prevent the use of a subquery? ______________________________

Topic #2: Subquery – Where Clause [SOME | ANY | ALL]

When working with a subquery within a Where clause you are able to use operators (**Some**, **Any**, **All)** if a comparison operator (=, >, <, >=, <=, < >) is used within the Where clause of the outer query.

Customers table

CustomerID	CustomerName
1	John Smith
2	Sue Doe
3	Dave Brown
4	Tina Jones
5	Tom Jackson

Orders table

OrderID	Date	CustomerID	Total
1	02/22/18	1	100
2	02/26/18	1	50
3	03/2/18	3	20
4	03/12/18	3	50
5	04/3/18	3	70
6	05/30/18	5	100
7	06/22/18	5	150

Example #1

Select CustomerName, Date, Total

From Customers Join Orders On Customers.CustomerID=Orders.CustomerID

Where **Total > ALL** (Select Total from Orders Where CustomerID=3)

Values returned from the subquery……………**(20, 50, 70)** The outer query will return records whose customer total is greater than all values returned in the subquery, which means, it will be **greater than its largest total**, 70.

Example #2

Select CustomerName, Date, Total

From Customers Join Orders On Customers.CustomerID=Orders.CustomerID

Where **Total < ALL** (Select Total from Orders Where CustomerID=3)

Values returned from subquery…………………**(20, 50, 70)** The outer query will return records whose customer total is less than all values returned in the subquery, which means, it will be **less than the smallest total**, 20.

Cis 111 Chapter 6 Student Lecture Notes

Customers table

CustomerID	CustomerName
1	John Smith
2	Sue Doe
3	Dave Brown
4	Tina Jones
5	Tom Jackson

Orders table

OrderID	Date	CustomerID	Total
1	02/22/18	1	100
2	02/26/18	1	50
3	03/2/18	3	20
4	03/12/18	3	50
5	04/3/18	3	70
6	05/30/18	5	100
7	06/22/18	5	150

Example #3

Select CustomerName, Date, Total

From Customers Join Orders On Customers.CustomerID=Orders.CustomerID

Where **Total > ANY** (Select Total from Orders Where CustomerID=3)

Values returned from the subquery…………..**(20, 50, 70)** The outer query will return records whose customer total is greater than any values returned in the subquery, which means, it will be **greater than its smallest total**, 20.

Example #4

Select CustomerName, Date, Total

From Customers Join Orders On Customers.CustomerID=Orders.CustomerID

Where **Total < ANY** (Select Total from Orders Where CustomerID=3)

Values returned from the subquery…………..**(20, 50, 70)** The outer query will return records whose customer total is greater than any values returned in the subquery, which means, it will be **less than its largest total**, 70.

Example #5

Select CustomerName, Date, Total

From Customers Join Orders On Customers.CustomerID=Orders.CustomerID

Where **Total = ANY** (Select Total from Orders Where CustomerID=3)

Values returned from the subquery…………..**(20, 50, 70)** The outer query will return records whose customer total is equal to any of the values returned in the subquery, which means, it will be **Total=20 or Total=50 or Total=70**

Lab Activity

6. Create a subquery for exercise #3 on page 212.

Topic #3: A Correlated Subquery

The subquery examples you have reviewed so far always had the subquery run completely first to either return a single value or a column of values. The data from the subquery would then be used in a Where clause of an outer query. There are times when a subquery needs to be executed once for each row in an outer query. This is referred to as a **correlated subquery**.

Customers table

CustomerID	CustomerName
1	John Smith
2	Sue Doe
3	Dave Brown
4	Tina Jones
5	Tom Jackson

Orders table

OrderID	Date	CustomerID	Total
1	02/22/18	1	100
2	02/26/18	1	50
3	03/2/18	3	20
4	03/12/18	3	50
5	04/3/18	3	70
6	05/30/18	5	100
7	06/22/18	5	150

Example #1

```
Select *
        From Orders
        Where Total > (Select AVG(Total)
                          From Orders As Orders_Sub
                              Where Orders_Sub.CustomerID = Orders.CustomerID)
```

How many rows would exist in the outer query? _______________

How many times would the subquery be executed? _____________

How would the first row in the outer query be processed? Please review the following example:

Orders

OrderID	Date	CustomerID	Total
1	02/22/18	1	100
2	02/26/18	1	50
3	03/2/18	3	20
4	03/12/18	3	50
5	04/3/18	3	70
6	05/30/18	5	100
7	06/22/18	5	150

The subquery will be executed for the first row which will return the average total for customer 1 which is **75**. Since the Total column for row 1 is 100, this row will be included in the result set. The next row in the outer query will go through the same process.

Which rows in the outer query will be included in the final result set?

Rows: ______________________________

Helpful info: Customer 1 average is 75, Customer 3 is 46.67 while Customer 5 is 125.

Example #2

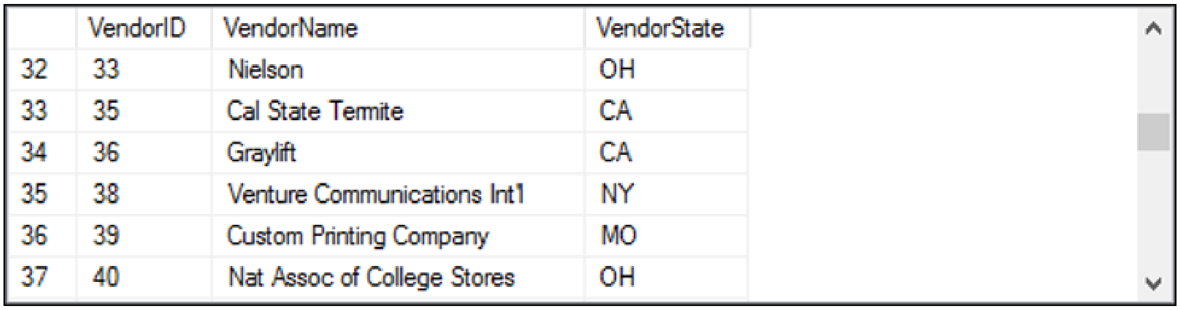
An **Exists operator** can be used in a correlated subquery to test the existence of any record in a subquery. The Exists operator returns true if the subquery returns one or more records.

A query that returns vendors without invoices

```
SELECT VendorID, VendorName, VendorState
FROM Vendors
WHERE NOT EXISTS
    (SELECT *
    FROM Invoices
    WHERE Invoices.VendorID = Vendors.VendorID);
```

Not Exists indicates that the result set will contain records from the outer query that were **not found** in the subquery.

The result set



	VendorID	VendorName	VendorState
32	33	Nielson	OH
33	35	Cal State Termite	CA
34	36	Graylift	CA
35	38	Venture Communications Int'l	NY
36	39	Custom Printing Company	MO
37	40	Nat Assoc of College Stores	OH

```
(88 rows)
```

How can you tell if a Select statemen is coded with a correlated subquery? The Where clause introduced in the _______________ compares a field from the _____________ to the ______________ query.

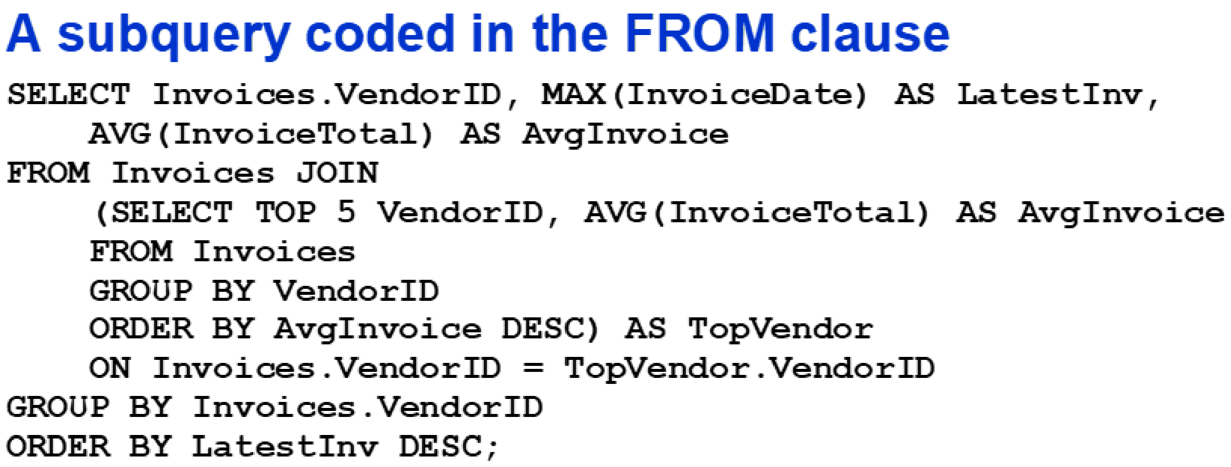
Lab Activity

7. Create a subquery for exercise #4 on page 213.
8. Create a subquery for exercise #5 on page 213.

Topic #4: Subquery Coded in the From Clause

When **subqueries** are used in the **FROM clause** they act as an interim table that you can use to select columns and join to other tables. Because of this some people argue they really aren't subqueries, but are **derived tables**, which are a special case of subqueries…subqueries used in the FROM clause!

Example #1



A subquery coded in the FROM clause

```
SELECT Invoices.VendorID, MAX(InvoiceDate) AS LatestInv,
    AVG(InvoiceTotal) AS AvgInvoice
FROM Invoices JOIN
    (SELECT TOP 5 VendorID, AVG(InvoiceTotal) AS AvgInvoice
    FROM Invoices
    GROUP BY VendorID
    ORDER BY AvgInvoice DESC) AS TopVendor
    ON Invoices.VendorID = TopVendor.VendorID
GROUP BY Invoices.VendorID
ORDER BY LatestInv DESC;
```

Items to point out in the above example:

1. The basic syntax of joining a table to a subquery is

   From Table1 **Join** (subquery) As AliasName **On** Table1.commonfield=AliasName.commonfield

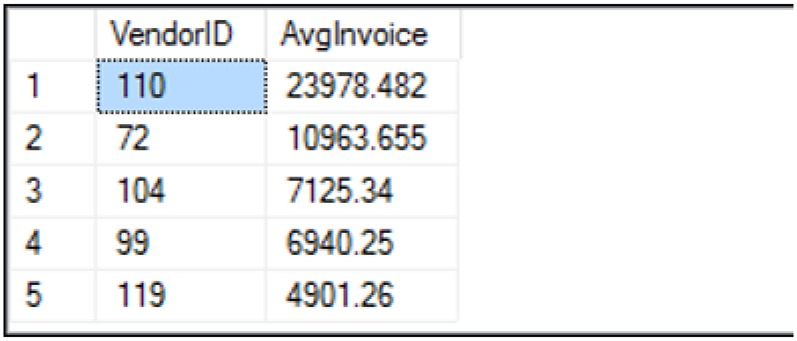
   You know that joining tables usually means a common field exists between the two so make sure the subquery returns a column that is common to a field in the table.

2. Columns that are defined in the subquery (derived table) can be referenced in the outer query.

The subquery would return the following derived table (TopVendor)

```
(SELECT TOP 5 VendorID, AVG(InvoiceTotal) AS
AvgInvoice
    FROM Invoices
    GROUP BY VendorID
    ORDER BY AvgInvoice DESC) AS TopVendor
```

TopVendor

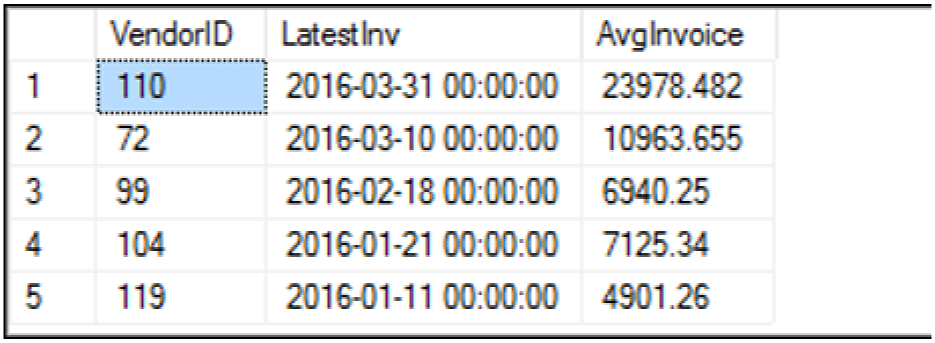


	VendorID	AvgInvoice
1	110	23978.482
2	72	10963.655
3	104	7125.34
4	99	6940.25
5	119	4901.26

Now SQL Server will join the Invoices table to TopVendor

```
SELECT Invoices.VendorID, MAX(InvoiceDate) AS
LatestInv, AVG(InvoiceTotal) AS AvgInvoice
FROM Invoices JOIN TopVendor
    ON Invoices.VendorID = TopVendor.VendorID
GROUP BY Invoices.VendorID
ORDER BY LatestInv DESC;
```

Final Result Set



	VendorID	LatestInv	AvgInvoice
1	110	2016-03-31 00:00:00	23978.482
2	72	2016-03-10 00:00:00	10963.655
3	99	2016-02-18 00:00:00	6940.25
4	104	2016-01-21 00:00:00	7125.34
5	119	2016-01-11 00:00:00	4901.26

Lab Activity

3. Create a subquery for exercise #6 on page 213.
4. Create a solution for exercise #7 on page 213.
5. Create a SQL statement for exercise #8 on page 213.

Topic #5: Subquery Coded in the Select Clause

A subquery can be introduced in a Select clause which means the subquery must return a single column or value. You will now review a couple of examples:

Customers table

CustomerID	CustomerName
1	John Smith
2	Sue Doe
3	Dave Brown
4	Tina Jones
5	Tom Jackson

⇔

Orders table

OrderID	Date	CustomerID	Total
1	02/22/18	1	100
2	02/26/18	1	50
3	03/2/18	3	20
4	03/12/18	3	50
5	04/3/18	3	70
6	05/30/18	5	100
7	06/22/18	5	150

Example #1

```
Select CustomerName, (Select Count(*) From Orders
                     Where Orders.CustomerID=Cust.CustomerID) as OrderCount
      From Customers As Cust
      Order by CustomerName
```

Complete the following result set that would be returned from the above query:

CustomerName	OrderCount

How do you know if the subquery in example #1 is a correlated subquery?

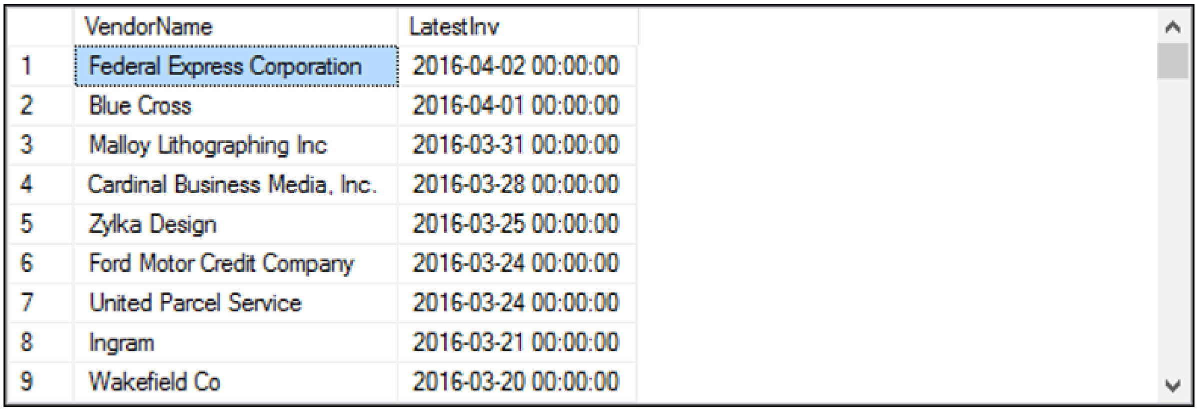
____

____

Example #2

```
SELECT DISTINCT VendorName,
     (SELECT MAX(InvoiceDate) FROM Invoices
     WHERE Invoices.VendorID = Vendors.VendorID) AS
LatestInv
FROM Vendors
ORDER BY LatestInv DESC;
```

The result set



	VendorName	LatestInv
1	Federal Express Corporation	2016-04-02 00:00:00
2	Blue Cross	2016-04-01 00:00:00
3	Malloy Lithographing Inc	2016-03-31 00:00:00
4	Cardinal Business Media, Inc.	2016-03-28 00:00:00
5	Zylka Design	2016-03-25 00:00:00
6	Ford Motor Credit Company	2016-03-24 00:00:00
7	United Parcel Service	2016-03-24 00:00:00
8	Ingram	2016-03-21 00:00:00
9	Wakefield Co	2016-03-20 00:00:00

(122 rows)

Cis 111 Chapter 6 Student Lecture Notes

Topic #6: Common Table Expressions (CTE)

Queries can become very complex at times so a developer can make use of a **common table expression** to simply the SQL syntax. You can think of a CTE as a derived table that is defined so that a following Select statement can use it. The example you will review is not complex but will be used to understand the syntax of a CTE

Example #1

```
SELECT      VendorName, Vendors.VendorState, VendorContactLName, VendorContactFName,
VendorPhone,  MidwestVendors.[Average Invoice Totals],MidwestVendors.[Invoice
Totals],MidwestVendors.[Number of Invoices]  FROM  Vendors Join
subquery
   (SELECT       Invoices.VendorID, Vendors.VendorState, SUM(Invoices.InvoiceTotal) AS
                 [Invoice Totals],  AVG(Invoices.InvoiceTotal) AS [Average Invoice Totals],
                 COUNT(Invoices.InvoiceTotal) AS [Number of Invoices]
     FROM   Invoices INNER JOIN Vendors ON Invoices.VendorID = Vendors.VendorID
     WHERE  (Vendors.VendorState IN ('IA', 'IL', 'KS', 'MI', 'MN', 'MO', 'OH', 'WI'))
     GROUP BY Invoices.VendorID, Vendors.VendorState
) As MidwestVendors On Vendors.VendorID=MidwestVendors.VendorID
ORDER BY MidwestVendors.[Invoice Totals] DESC
```

The next example will define the above subquery using a CTE:

```
With MidwestVendors As
(SELECT  Invoices.VendorID, VendorState,
           SUM(InvoiceTotal) AS [Invoice Totals],
           AVG(InvoiceTotal) AS [Average Invoice Totals],                  C
           COUNT(InvoiceTotal)  AS [Number of Invoices]                    T
FROM     Invoices INNER JOIN  Vendors ON Invoices.VendorID = Vendors.VendorID
WHERE    (VendorState IN ('IA', 'IL', 'KS', 'MI', 'MN', 'MO', 'OH', 'WI'))     E
GROUP BY Invoices.VendorID, VendorState
)
SELECT   VendorName, Vendors.VendorState, VendorContactLName, VendorContactFName,
           VendorPhone, MidwestVendors.[Average Invoice Totals],
            MidwestVendors.[Invoice Totals],MidwestVendors.[Number of Invoices]
FROM   Vendors Join MidwestVendors ON Vendors.VendorID=MidwestVendors.VendorID
Order by MidwestVendors.[Invoice Totals] Desc
```

You can also define the column names for a CTE. Here is the same solution with defining specific column names for the CTE:

```
With MidwestVendors(ID,State,InvTotal,AvgInvoice, InvCount) As
(
SELECT    Invoices.VendorID, VendorState, SUM(InvoiceTotal) AS [Invoice Totals],
          AVG(Invoices.InvoiceTotal) AS [Average Invoice Totals],
          COUNT(Invoices.InvoiceTotal)  AS [Number of Invoices]
FROM      Invoices INNER JOIN  Vendors ON Invoices.VendorID = Vendors.VendorID
WHERE     (VendorState IN ('IA', 'IL', 'KS', 'MI', 'MN', 'MO', 'OH', 'WI'))
GROUP BY Invoices.VendorID, VendorState
)

SELECT    VendorName, Vendors.VendorState, VendorContactLName, VendorContactFName,
          VendorPhone, MidwestVendors.AvgInvoice, MidwestVendors.InvTotal,
          MidwestVendors.InvCount
FROM      Vendors Join MidwestVendors ON Vendors.VendorID=MidwestVendors.ID
Order by MidwestVendors.InvCount Desc
```

Lab Activity

6. Create a solution using a CTE for exercise #9 on page 213.