LAB 08

SUBQUERY

❖ **Main contents:** Concept and use of subqueries, correlated and non-correlated subqueries.

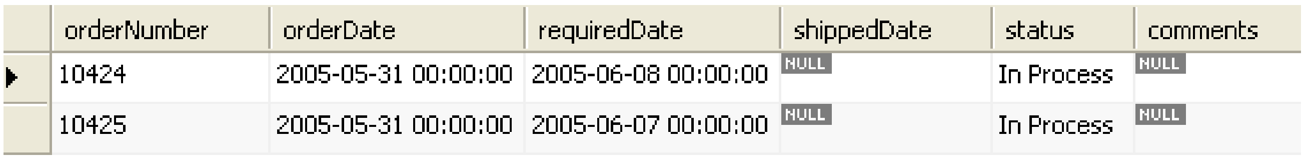
1. Concept of Subquery

To combine data tables together, in addition to joins and set operators, SQL provides another way to return data from multiple tables called subquery. When one `SELECT` statement is used in another, the inner `SELECT` is called a *subquery*, another way is called a *nested query*, or an *inner query*. Basically, a subquery can be used anywhere an expression can be used.

Example: Give the most recent orders

```
SELECT * FROM orders
WHERE orderDate = (SELECT MAX(orderDate) FROM orders)
```

`SELECT MAX(orderDate) FROM orders` return the most recent date in the orders and this value will be used in the `WHERE` clause of the outer query. Combining the two queries above will return a list of orders for the most recent day.



orderNumber	orderDate	requiredDate	shippedDate	status	comments
10424	2005-05-31 00:00:00	2005-06-08 00:00:00	NULL	In Process	NULL
10425	2005-05-31 00:00:00	2005-06-07 00:00:00	NULL	In Process	NULL

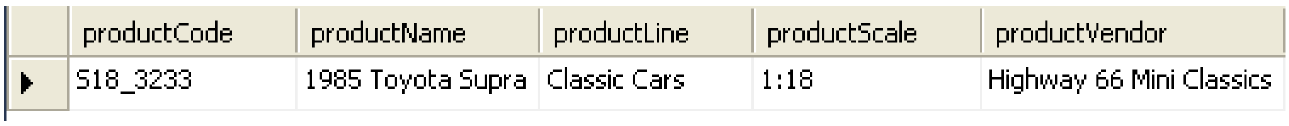
Subqueries are divided into two categories: non-correlated subqueries and correlated subqueries.

2. Non-correlated subquery

A non-correlated subquery is a subquery independent of an external query. The non-correlated subqueries are executed first and only once for the entire statement. The result of the subquery is populated with the outer query, and finally the external query is executed.

Example: Get products that are not included in any orders. The inner subquery will return the product codes included in the *orderdetails* table. The external subquery will return products whose codes are not in the list of product codes.

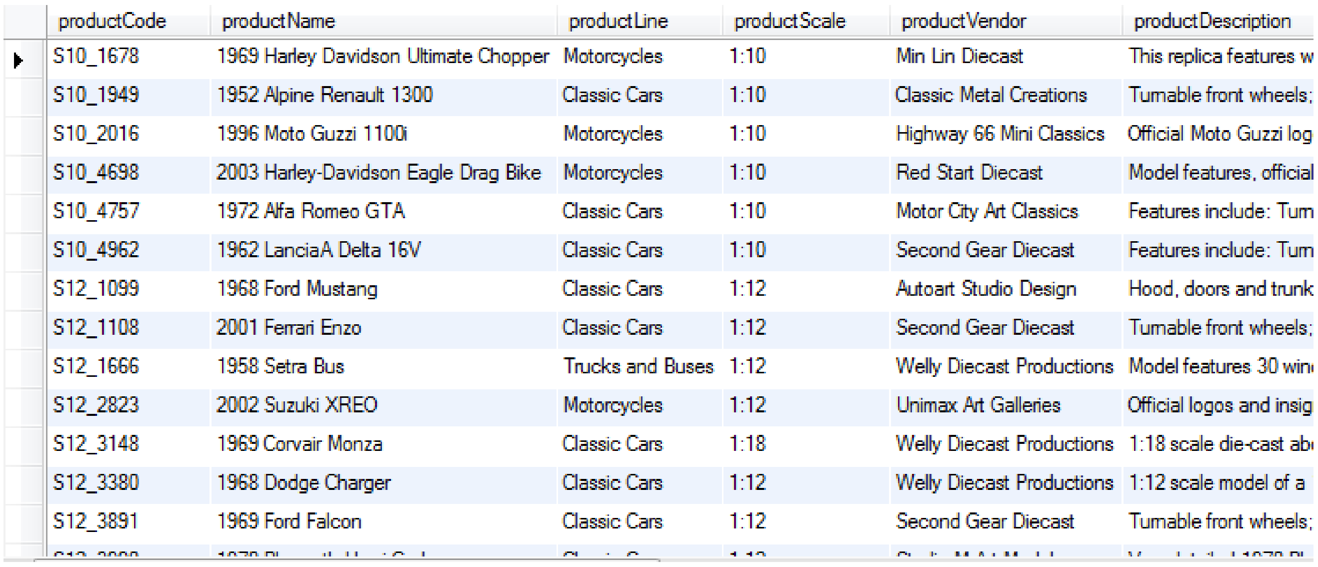
```
SELECT *
FROM products
WHERE productCode NOT IN
     (SELECT productCode
     FROM orderdetails
     );
```



	productCode	productName	productLine	productScale	productVendor
▶	S18_3233	1985 Toyota Supra	Classic Cars	1:18	Highway 66 Mini Classics

Example: Get the products that are included in orders

```
SELECT * FROM products
WHERE productCode IN
    (SELECT productCode
     FROM orderdetails
     );
```



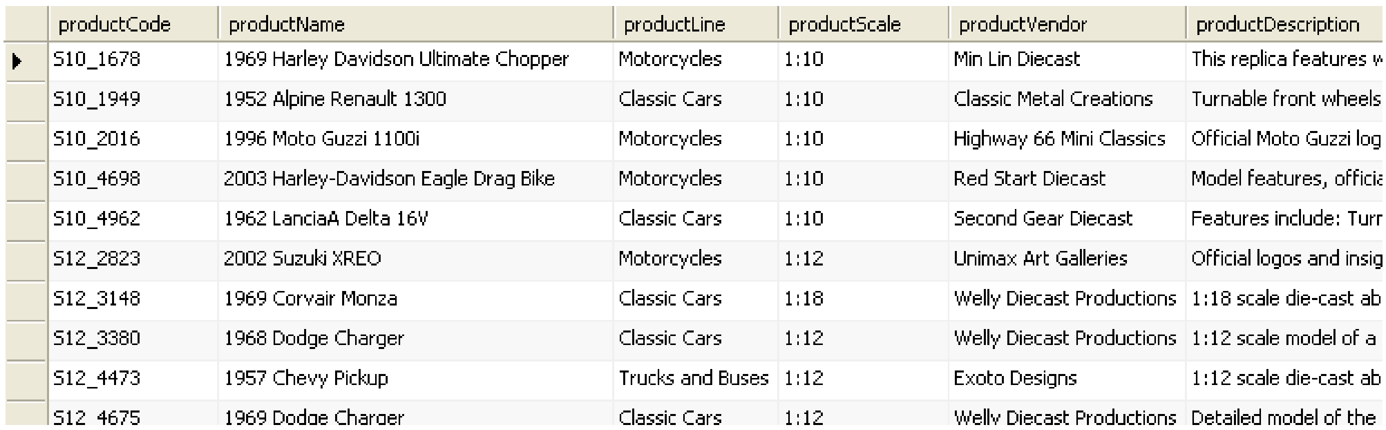
	productCode	productName	productLine	productScale	productVendor	productDescription
▶	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features w
	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels;
	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi log
	S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, official
	S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	Features include: Turn
	S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Turn
	S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design	Hood, doors and trunk
	S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels;
	S12_1666	1958 Setra Bus	Trucks and Buses	1:12	Welly Diecast Productions	Model features 30 win
	S12_2823	2002 Suzuki XREO	Motorcycles	1:12	Unimax Art Galleries	Official logos and insig
	S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Productions	1:18 scale die-cast ab
	S12_3380	1968 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	1:12 scale model of a
	S12_3891	1969 Ford Falcon	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels;

3. Correlated subquery

A correlated subquery is not independent of the external query. A correlated subquery is a subquery that uses values from an external query in its `WHERE` clause. The process is as follows: external queries are executed first and then inner subqueries are executed for each result of the external queries.

Example: Get the products with a quantity in stock greater than the average quantity of products of the same product line.

```
SELECT * FROM products p
WHERE quantityInStock >
     (SELECt avg(quantityInStock)
      FROM products
      WHERE productLine = p.productLine
     );
```

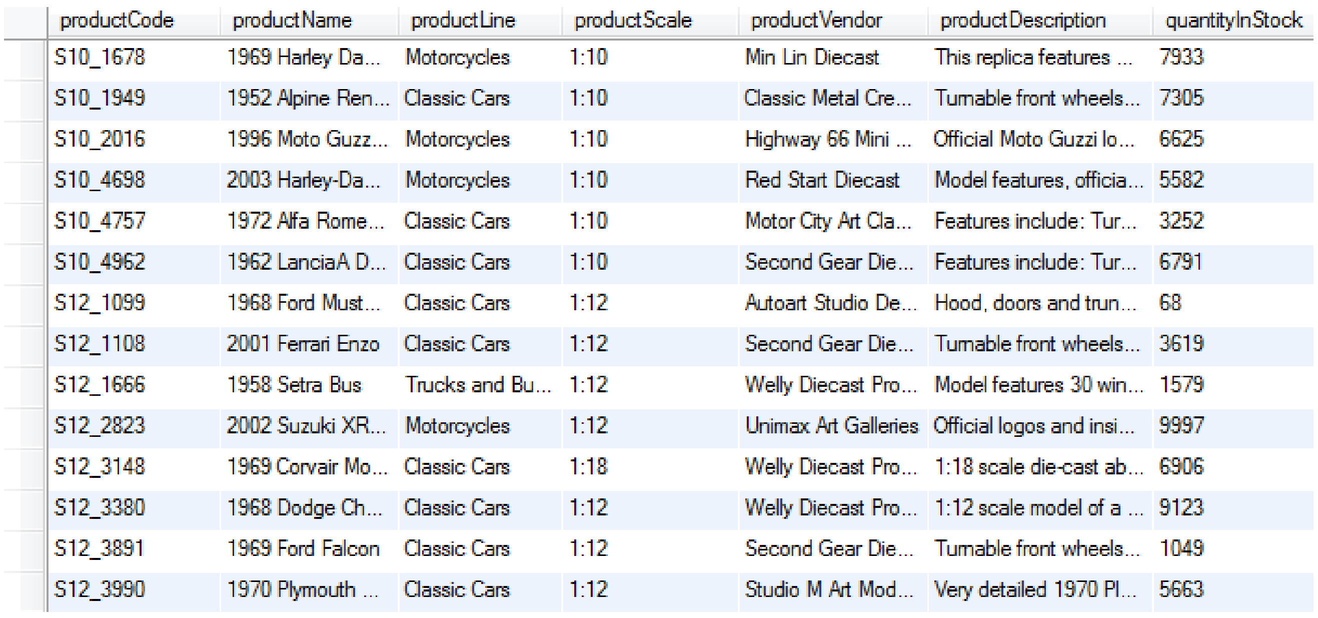


productCode	productName	productLine	productScale	productVendor	productDescription
S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features w
S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels
S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi log
S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, officia
S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Turr
S12_2823	2002 Suzuki XREO	Motorcycles	1:12	Unimax Art Galleries	Official logos and insig
S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Productions	1:18 scale die-cast ab
S12_3380	1968 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	1:12 scale model of a
S12_4473	1957 Chevy Pickup	Trucks and Buses	1:12	Exoto Designs	1:12 scale die-cast ab
S12_4675	1969 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	Detailed model of the

The query execution process is as follows: for each product line of an external query, the internal query statement finds the average quantity of products of the same product line. The subquery will be put into the `WHERE` clause to examine.

Example: Get the products included in the order. Use the `EXISTS` operator to check for existence.

```
SELECT * FROM products as p
WHERE exists
          (SELECT productCode
           FROM orderdetails
           WHERE productCode = p.productCode);
```



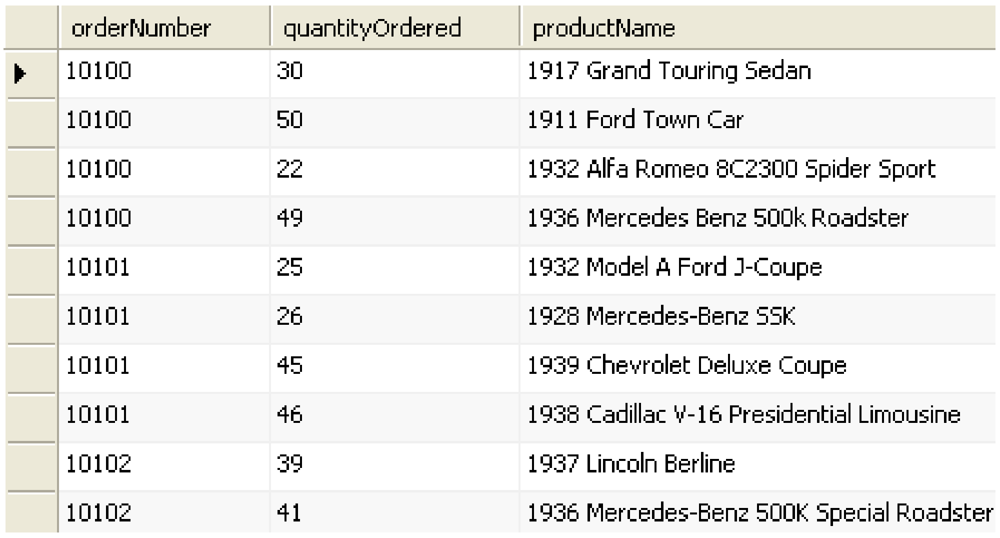
productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock
S10_1678	1969 Harley Da...	Motorcycles	1:10	Min Lin Diecast	This replica features ...	7933
S10_1949	1952 Alpine Ren...	Classic Cars	1:10	Classic Metal Cre...	Turnable front wheels...	7305
S10_2016	1996 Moto Guzz...	Motorcycles	1:10	Highway 66 Mini ...	Official Moto Guzzi lo...	6625
S10_4698	2003 Harley-Da...	Motorcycles	1:10	Red Start Diecast	Model features, officia...	5582
S10_4757	1972 Alfa Rome...	Classic Cars	1:10	Motor City Art Cla...	Features include: Tur...	3252
S10_4962	1962 LanciaA D...	Classic Cars	1:10	Second Gear Die...	Features include: Tur...	6791
S12_1099	1968 Ford Must...	Classic Cars	1:12	Autoart Studio De...	Hood, doors and trun...	68
S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Die...	Turnable front wheels...	3619
S12_1666	1958 Setra Bus	Trucks and Bu...	1:12	Welly Diecast Pro...	Model features 30 win...	1579
S12_2823	2002 Suzuki XR...	Motorcycles	1:12	Unimax Art Galleries	Official logos and insi...	9997
S12_3148	1969 Corvair Mo...	Classic Cars	1:18	Welly Diecast Pro...	1:18 scale die-cast ab...	6906
S12_3380	1968 Dodge Ch...	Classic Cars	1:12	Welly Diecast Pro...	1:12 scale model of a ...	9123
S12_3891	1969 Ford Falcon	Classic Cars	1:12	Second Gear Die...	Turnable front wheels...	1049
S12_3990	1970 Plymouth ...	Classic Cars	1:12	Studio M Art Mod...	Very detailed 1970 Pl...	5663

4. Use subqueries

In addition to using subqueries in the `WHERE` clause, subqueries can also be used in the list of columns of the `SELECT` statement or in the `FROM` clause.

Example: For each order line, include the name of the product.

```
SELECT orderNumber, quantityOrdered,
     (SELECT productName FROM products WHERE productCode =
     o.productCode) as productName
FROM orderdetails o;
```

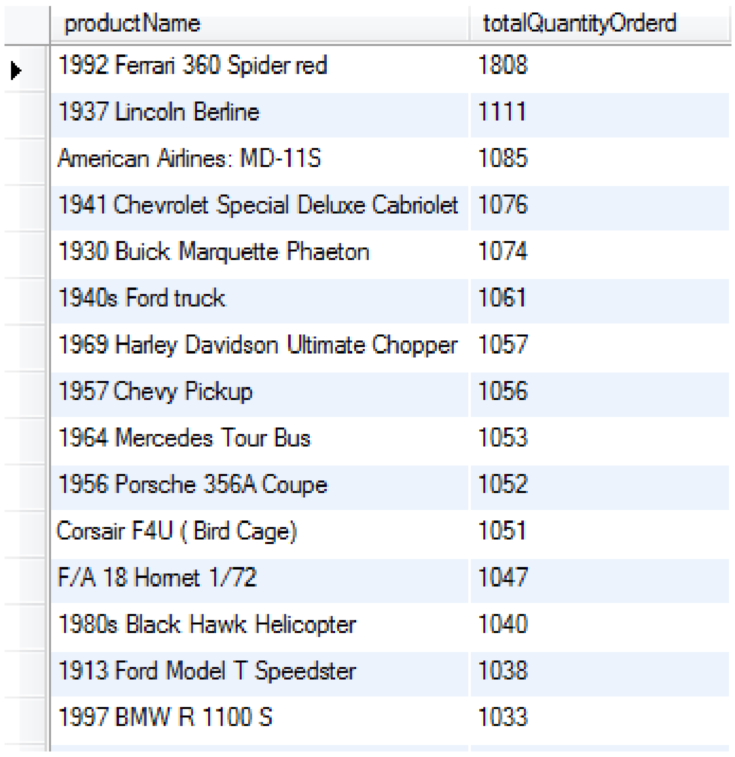


orderNumber	quantityOrdered	productName
10100	30	1917 Grand Touring Sedan
10100	50	1911 Ford Town Car
10100	22	1932 Alfa Romeo 8C2300 Spider Sport
10100	49	1936 Mercedes Benz 500k Roadster
10101	25	1932 Model A Ford J-Coupe
10101	26	1928 Mercedes-Benz SSK
10101	45	1939 Chevrolet Deluxe Coupe
10101	46	1938 Cadillac V-16 Presidential Limousine
10102	39	1937 Lincoln Berline
10102	41	1936 Mercedes-Benz 500K Special Roadster

In the example above the name of the product is the result of the subquery on the *products* table.

Example: For each product, include the total number of products that were ordered

```
SELECT productName,
     (SELECT sum(quantityOrdered) FROM orderdetails WHERE
     productCode = p.productCode) as totalQuantityOrderd
FROM products as p
ORDER BY totalQuantityOrderd desc
```



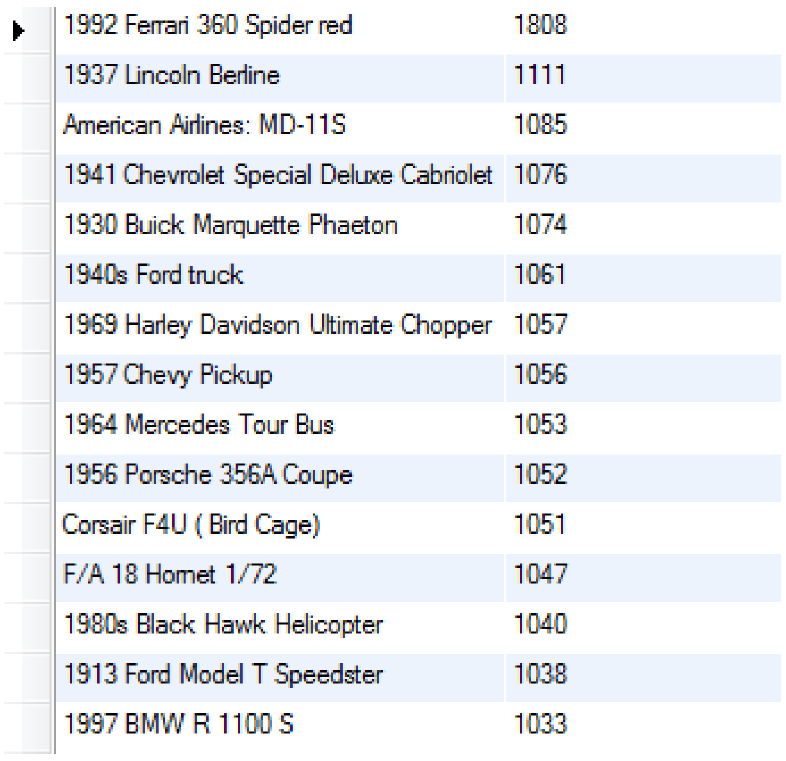
productName	totalQuantityOrderd
1992 Ferrari 360 Spider red	1808
1937 Lincoln Berline	1111
American Airlines: MD-11S	1085
1941 Chevrolet Special Deluxe Cabriolet	1076
1930 Buick Marquette Phaeton	1074
1940s Ford truck	1061
1969 Harley Davidson Ultimate Chopper	1057
1957 Chevy Pickup	1056
1964 Mercedes Tour Bus	1053
1956 Porsche 356A Coupe	1052
Corsair F4U (Bird Cage)	1051
F/A 18 Hornet 1/72	1047
1980s Black Hawk Helicopter	1040
1913 Ford Model T Speedster	1038
1997 BMW R 1100 S	1033

In the example above the total quantity value is set as the result of the query from the *orderDetails* table

The example above can be rewritten by treating the result of a subquery as a data table, and then connecting the *products* table to this result table.

```
SELECT productName, totalQuantityOrderd
FROM products,
(SELECT productCode, SUM(quantityOrdered) as
totalQuantityOrderd FROM  orderdetails group by productCode) AS
productOrder
WHERE products.productCode = productOrder.productCode;
```

The result of the query gives the same results as the previous query.



1992 Ferrari 360 Spider red	1808
1937 Lincoln Berline	1111
American Airlines: MD-11S	1085
1941 Chevrolet Special Deluxe Cabriolet	1076
1930 Buick Marquette Phaeton	1074
1940s Ford truck	1061
1969 Harley Davidson Ultimate Chopper	1057
1957 Chevy Pickup	1056
1964 Mercedes Tour Bus	1053
1956 Porsche 356A Coupe	1052
Corsair F4U (Bird Cage)	1051
F/A 18 Hornet 1/72	1047
1980s Black Hawk Helicopter	1040
1913 Ford Model T Speedster	1038
1997 BMW R 1100 S	1033

❖ **Practical Exercises:**

1. Use the subquery to list products that were ordered in March 2005. Compare to using JOIN instead of using subqueries.
2. Use the subquery to display information about orders in the most recent month (using information from the *orders* table).
3. Use subqueries to give information about orders and total value of this order (using information from *orders* and *orderdetails* tables). Compare to using JOIN instead of using subqueries.
4. Use the subquery get the customer's name and the total amount they have to pay.