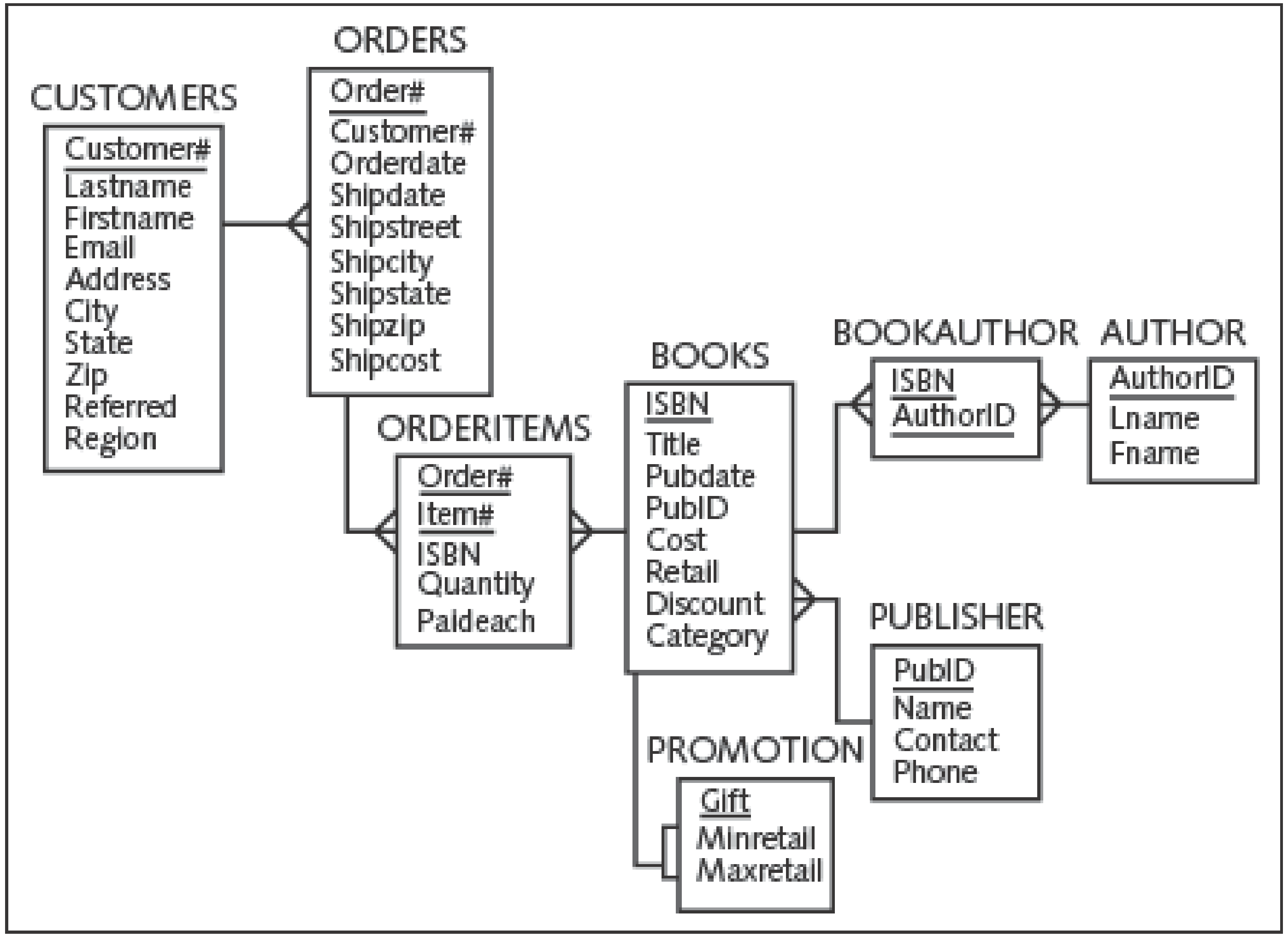
Database Systems
Joining Data From Multiple Tables

CS 630 Database Systems

Professor Nardi

KING GRADUATE SCHOOL MONROE COLLEGE

Normalized JustLee Books Database…



KING GRADUATE SCHOOL MONROE COLLEGE

Purpose of Joins…

- Used to Link Tables and Reconstruct Data in a Relational Database…
- Can Be Created Through:
  - Conditions in a WHERE Clause
  - Use of JOIN Keywords in the FROM Clause

KING GRADUATE SCHOOL MONROE COLLEGE

Cartesian Joins…

- Created By Omitting Joining Condition in the WHERE Clause or Through CROSS JOIN Keywords in the FROM Clause…
- Results In Every Possible Row Combination…
  - ✓ (m * n)…

For Example…

- Suppose You Wanted to Return a Report Showing Every Combination of Color and Size…

Tables

Color
Red
Blue

Size
Small
Medium
Large
Extra Large

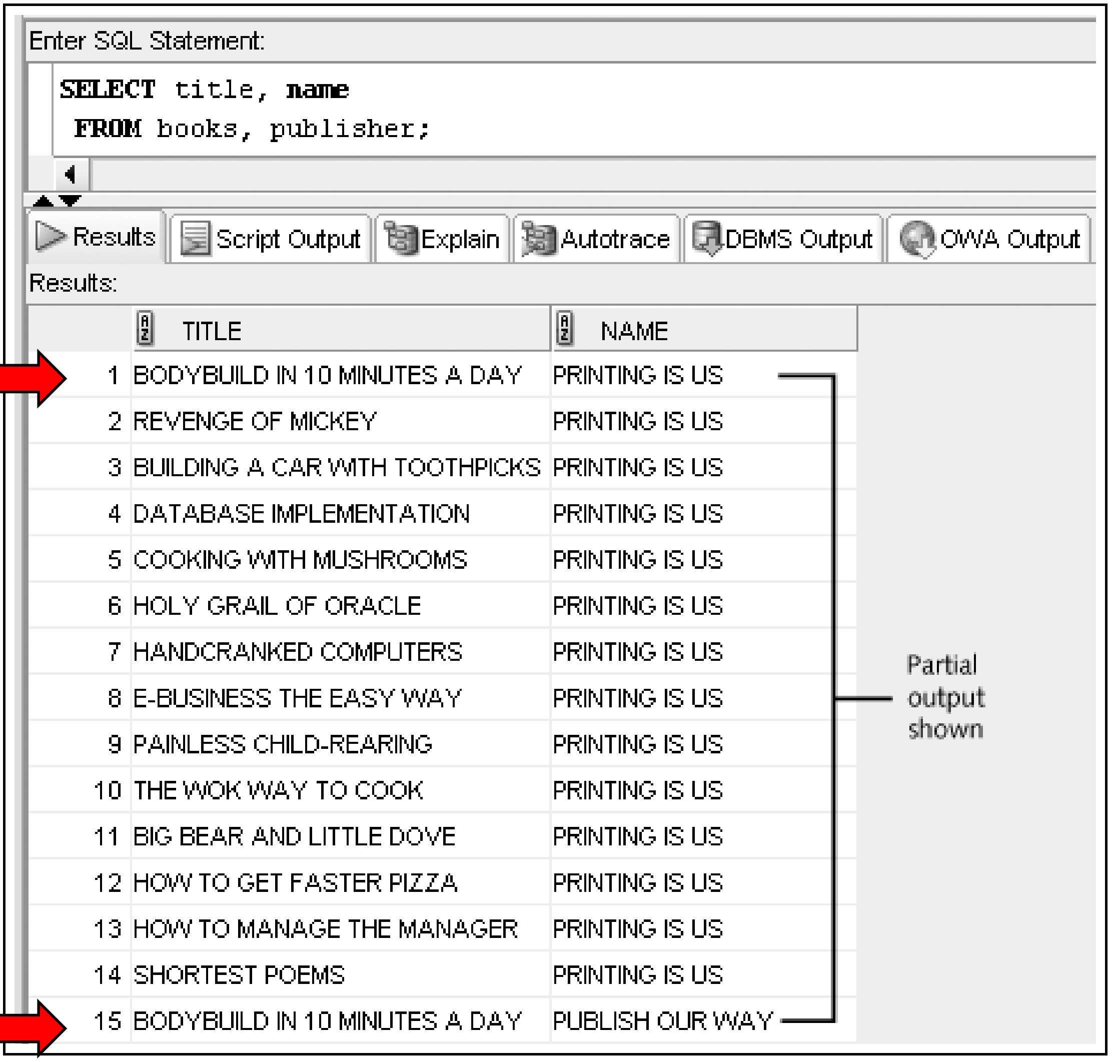
Query Result

Color	Size
Red	Small
Blue	Small
Red	Medium
Blue	Medium
Red	Large
Blue	Large
Red	Extra Large
Blue	Extra Large

KING GRADUATE SCHOOL MONROE COLLEGE

Cartesian Join Example - Omitted Condition…

- Look At the Highlighted Rows…
- See That Rows 1 Thru 14 Will Be Repeated for “PUBLISH OUR WAY”…
- It Will Do This for EVERY Publisher…



```
SELECT title, name
 FROM books, publisher;
```

	TITLE	NAME
1	BODYBUILD IN 10 MINUTES A DAY	PRINTING IS US
2	REVENGE OF MICKEY	PRINTING IS US
3	BUILDING A CAR WITH TOOTHPICKS	PRINTING IS US
4	DATABASE IMPLEMENTATION	PRINTING IS US
5	COOKING WITH MUSHROOMS	PRINTING IS US
6	HOLY GRAIL OF ORACLE	PRINTING IS US
7	HANDCRANKED COMPUTERS	PRINTING IS US
8	E-BUSINESS THE EASY WAY	PRINTING IS US
9	PAINLESS CHILD-REARING	PRINTING IS US
10	THE WOK WAY TO COOK	PRINTING IS US
11	BIG BEAR AND LITTLE DOVE	PRINTING IS US
12	HOW TO GET FASTER PIZZA	PRINTING IS US
13	HOW TO MANAGE THE MANAGER	PRINTING IS US
14	SHORTEST POEMS	PRINTING IS US
15	BODYBUILD IN 10 MINUTES A DAY	PUBLISH OUR WAY

KING GRADUATE SCHOOL MONROE COLLEGE

For Example…

- Suppose You Wanted to Return a Report Showing Every Combination of Color and Size…

SELECT color, size

FROM color, size;

Tables

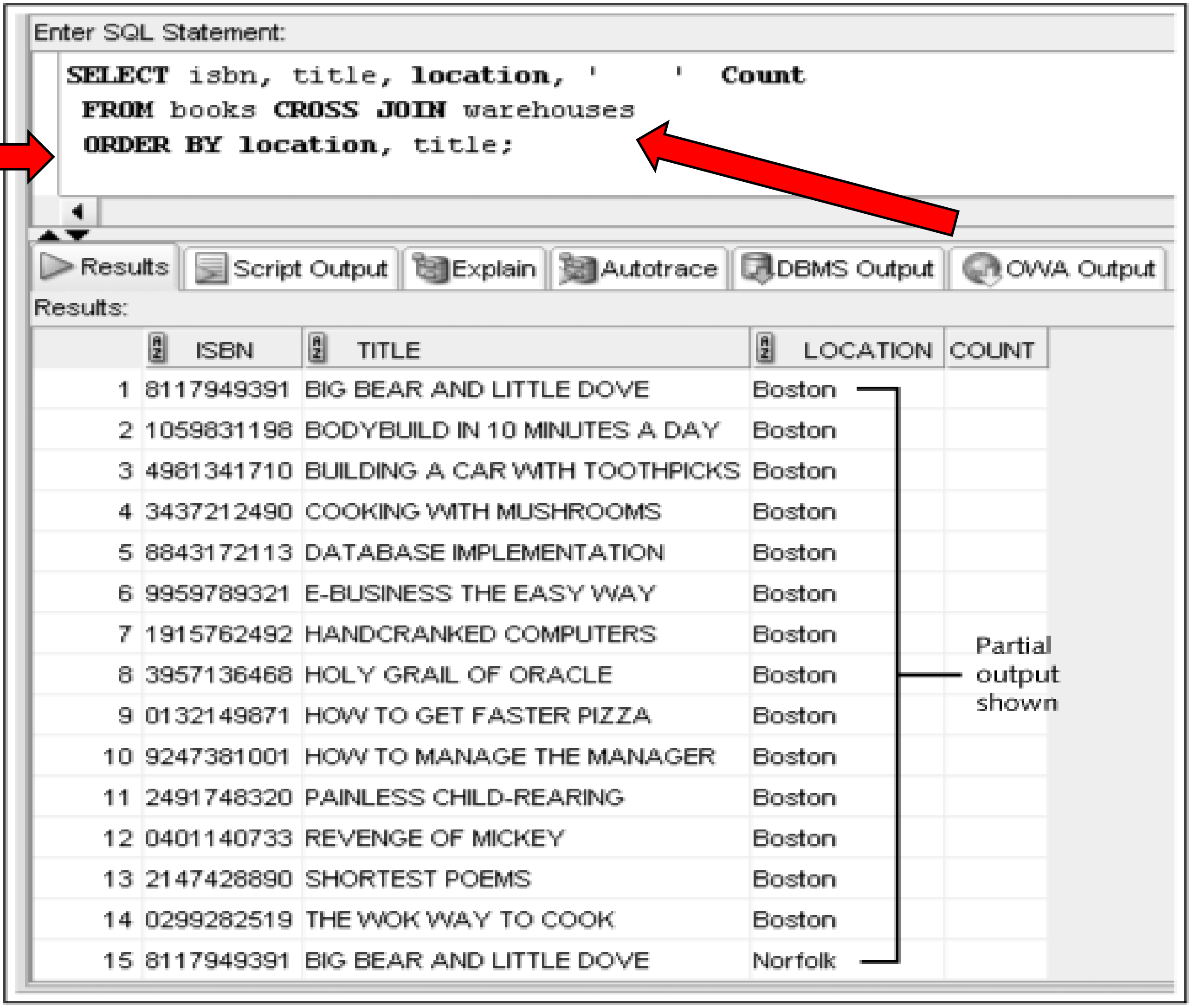
Color
Red
Blue

Size
Small
Medium
Large
Extra Large

Query Result	
Color	Size
Red	Small
Blue	Small
Red	Medium
Blue	Medium
Red	Large
Blue	Large
Red	Extra Large
Blue	Extra Large

Cartesian Join Example – CROSS JOIN…

- Also Produces a Cartesian Product…
- Notice There is NO JOIN Clause…
- CROSS JOINs Let You Specify a WHERE Clause…



```sql
SELECT isbn, title, location, '     ' Count
 FROM books CROSS JOIN warehouses
 ORDER BY location, title;
```

	ISBN	TITLE	LOCATION	COUNT
1	8117949391	BIG BEAR AND LITTLE DOVE	Boston	
2	1059831198	BODYBUILD IN 10 MINUTES A DAY	Boston	
3	4981341710	BUILDING A CAR WITH TOOTHPICKS	Boston	
4	3437212490	COOKING WITH MUSHROOMS	Boston	
5	8843172113	DATABASE IMPLEMENTATION	Boston	
6	9959789321	E-BUSINESS THE EASY WAY	Boston	
7	1915762492	HANDCRANKED COMPUTERS	Boston	
8	3957136468	HOLY GRAIL OF ORACLE	Boston	
9	0132149871	HOW TO GET FASTER PIZZA	Boston	
10	9247381001	HOW TO MANAGE THE MANAGER	Boston	
11	2491748320	PAINLESS CHILD-REARING	Boston	
12	0401140733	REVENGE OF MICKEY	Boston	
13	2147428890	SHORTEST POEMS	Boston	
14	0299282519	THE WOK WAY TO COOK	Boston	
15	8117949391	BIG BEAR AND LITTLE DOVE	Norfolk	

For Example…

- Suppose You Wanted to Return a Report Showing Every Combination of Color and Size…

SELECT color, size

FROM color CROSS JOIN size

WHERE color = ‘Red’;

Tables

Color
Red
Blue

Size
Small
Medium
Large
Extra Large

Query Result

Color	Size
Red	Small
[redacted]	[redacted]
Red	Medium
[redacted]	[redacted]
Red	Large
[redacted]	[redacted]
Red	Extra Large
[redacted]	[redacted]

KING GRADUATE SCHOOL MONROE COLLEGE

JOIN vs. CROSS JOIN…

- The following SELECT statements are equivalent:
  - ✓SELECT cities, airport
    FROM cities CROSS JOIN flights;
  - ✓SELECT cities, airport
    FROM cities, flights;
- Advantage CROSS JOIN…
  - ✓SELECT cities, airport
    FROM cities CROSS JOIN flights
    ORDER BY cities, airport;

KING GRADUATE SCHOOL MONROE COLLEGE

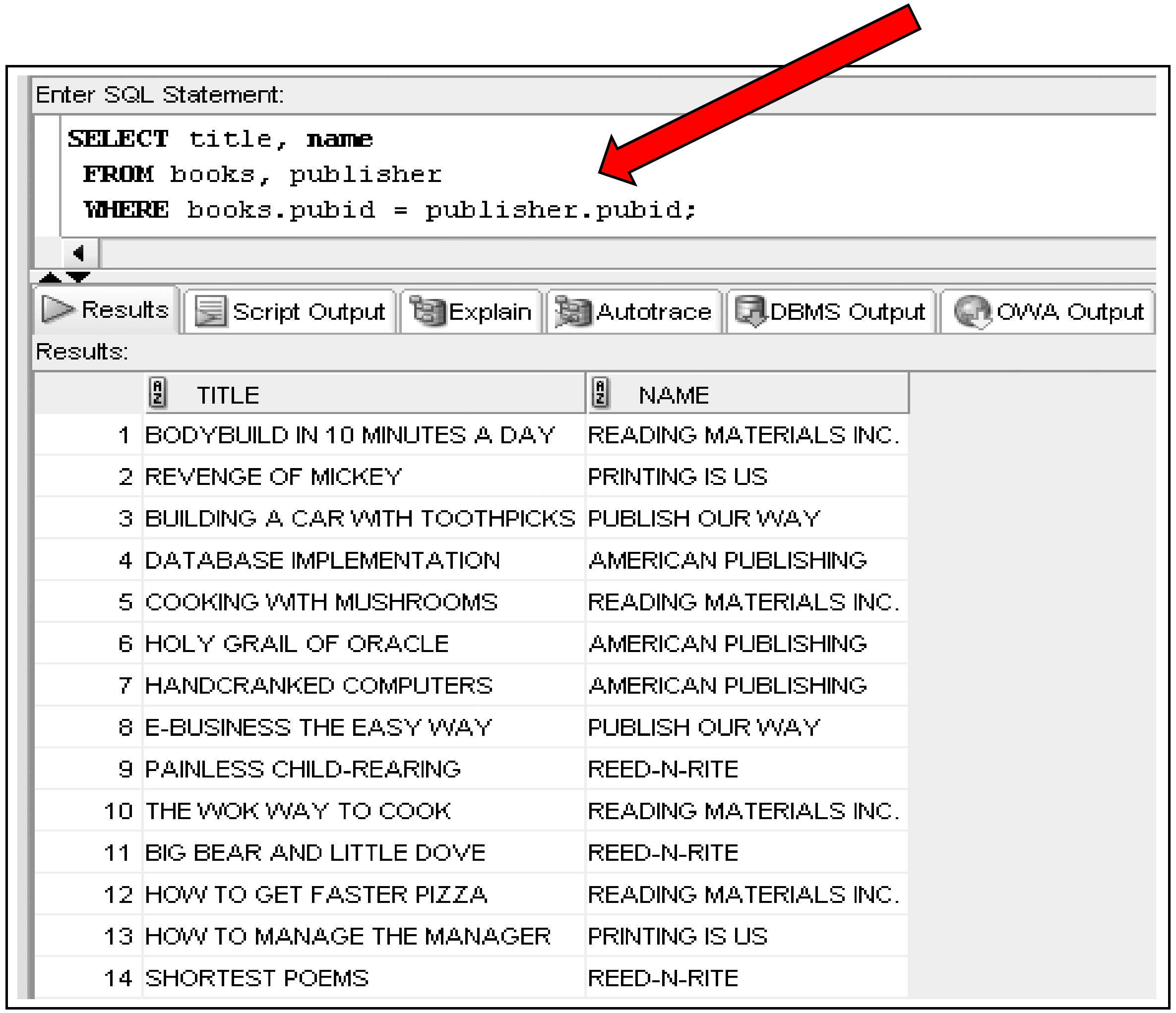
Equality Joins…

- Link Rows Through Equivalent Data That Exists in Both Tables…
- Created By an Equivalency Condition in the WHERE Clause…
- Can Use the Following Keyword in the FROM Clause:
  - NATURAL JOIN…
  - JOIN…USING…
  - JOIN…ON…

KING GRADUATE SCHOOL MONROE COLLEGE

Equality Joins: WHERE Clause Example…

- Notice the WHERE Clause Specifies BOTH the Table and the Element…
- When an Element Appears in More Than One Table in a JOIN, You Must Specify the Table With the Element…



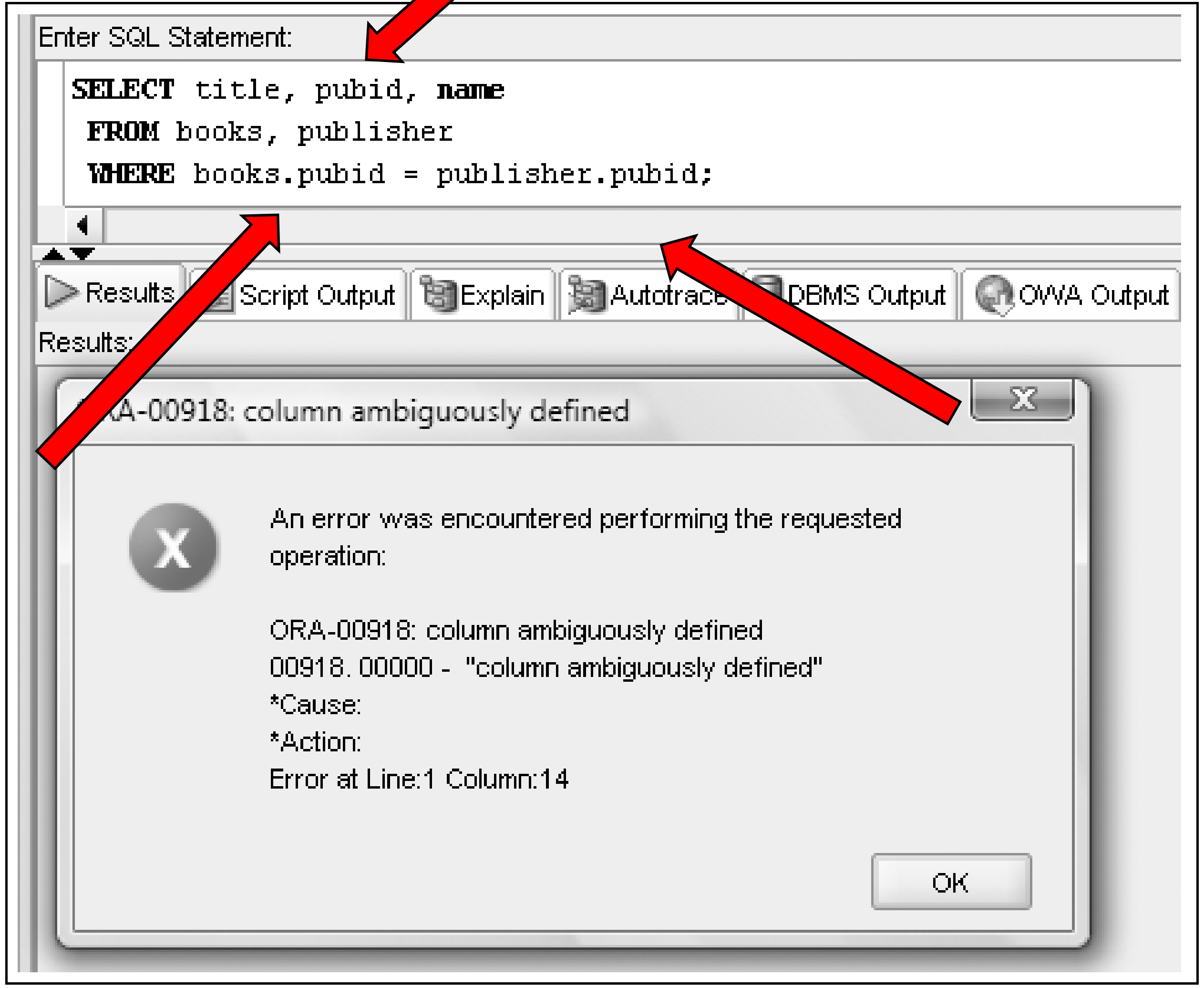
```
SELECT title, name
 FROM books, publisher
 WHERE books.pubid = publisher.pubid;
```

	TITLE	NAME
1	BODYBUILD IN 10 MINUTES A DAY	READING MATERIALS INC.
2	REVENGE OF MICKEY	PRINTING IS US
3	BUILDING A CAR WITH TOOTHPICKS	PUBLISH OUR WAY
4	DATABASE IMPLEMENTATION	AMERICAN PUBLISHING
5	COOKING WITH MUSHROOMS	READING MATERIALS INC.
6	HOLY GRAIL OF ORACLE	AMERICAN PUBLISHING
7	HANDCRANKED COMPUTERS	AMERICAN PUBLISHING
8	E-BUSINESS THE EASY WAY	PUBLISH OUR WAY
9	PAINLESS CHILD-REARING	REED-N-RITE
10	THE WOK WAY TO COOK	READING MATERIALS INC.
11	BIG BEAR AND LITTLE DOVE	REED-N-RITE
12	HOW TO GET FASTER PIZZA	READING MATERIALS INC.
13	HOW TO MANAGE THE MANAGER	PRINTING IS US
14	SHORTEST POEMS	REED-N-RITE

KING GRADUATE SCHOOL MONROE COLLEGE

Qualifying Column Names…

- Columns Appearing in More Than One Table Must Be Qualified in Both the SELECT and the WHERE…


Enter SQL Statement:

```
SELECT title, pubid, name
 FROM books, publisher
 WHERE books.pubid = publisher.pubid;
```

ORA-00918: column ambiguously defined

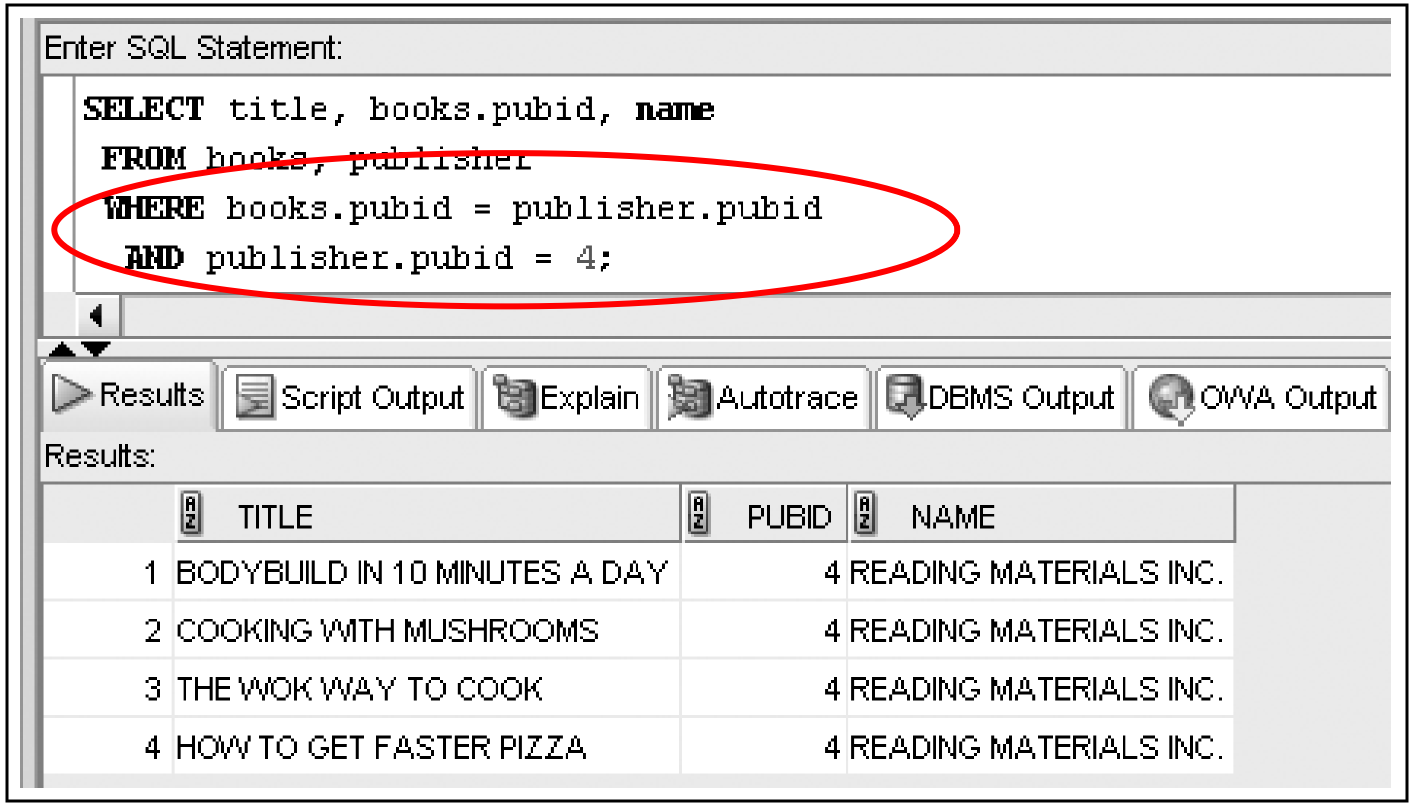
An error was encountered performing the requested operation:

ORA-00918: column ambiguously defined
00918. 00000 - "column ambiguously defined"
*Cause:
*Action:
Error at Line:1 Column:14

KING GRADUATE SCHOOL MONROE COLLEGE

WHERE Clause Supports Join and Other Conditions…

- Adding Search Conditions Helps Limit the Data Returned…
- Any of the Items We Discussed in the "Other Stuff" Presentation Can Be Used Here…


Enter SQL Statement:

```
SELECT title, books.pubid, name
 FROM books, publisher
 WHERE books.pubid = publisher.pubid
  AND publisher.pubid = 4;
```

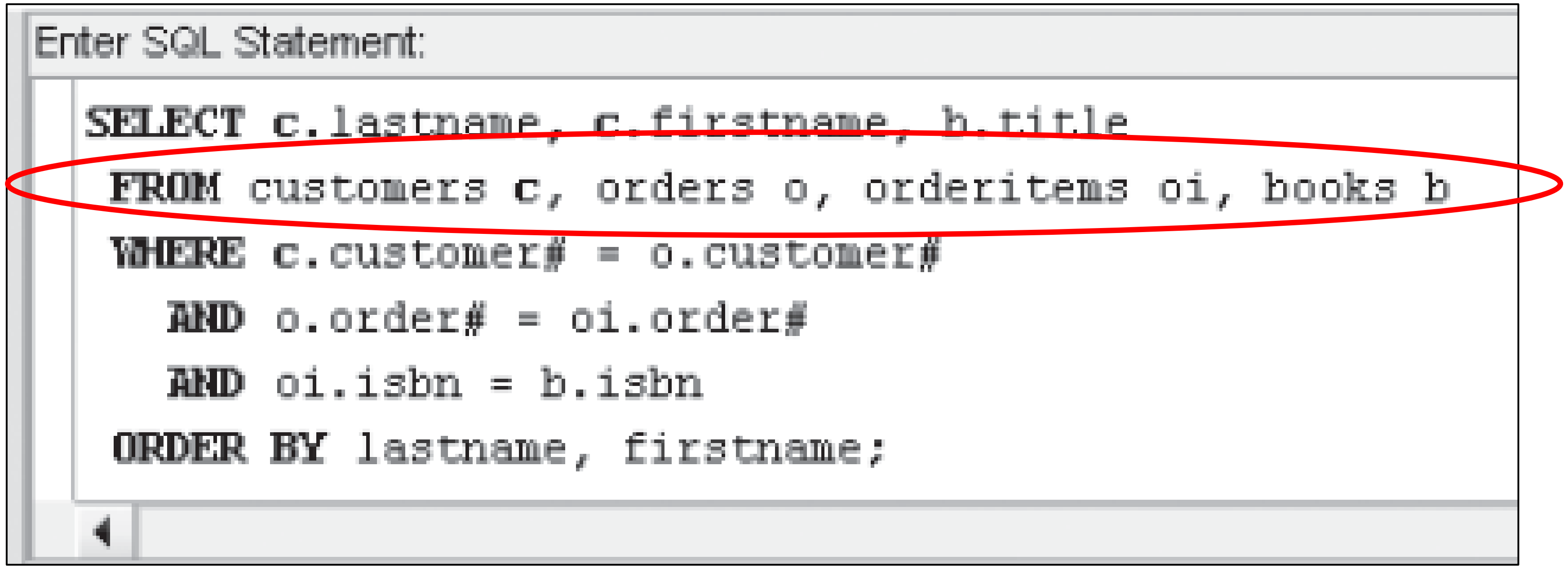
Results:

	TITLE	PUBID	NAME
1	BODYBUILD IN 10 MINUTES A DAY	4	READING MATERIALS INC.
2	COOKING WITH MUSHROOMS	4	READING MATERIALS INC.
3	THE WOK WAY TO COOK	4	READING MATERIALS INC.
4	HOW TO GET FASTER PIZZA	4	READING MATERIALS INC.

KING GRADUATE SCHOOL MONROE COLLEGE

Aliases and Joining More Than Two Tables…

- Joining Four Tables Requires Three JOIN Conditions…
- An Alias is a Shorten Version of the Table Name…
- The Alias is Defined in the FROM Clause…

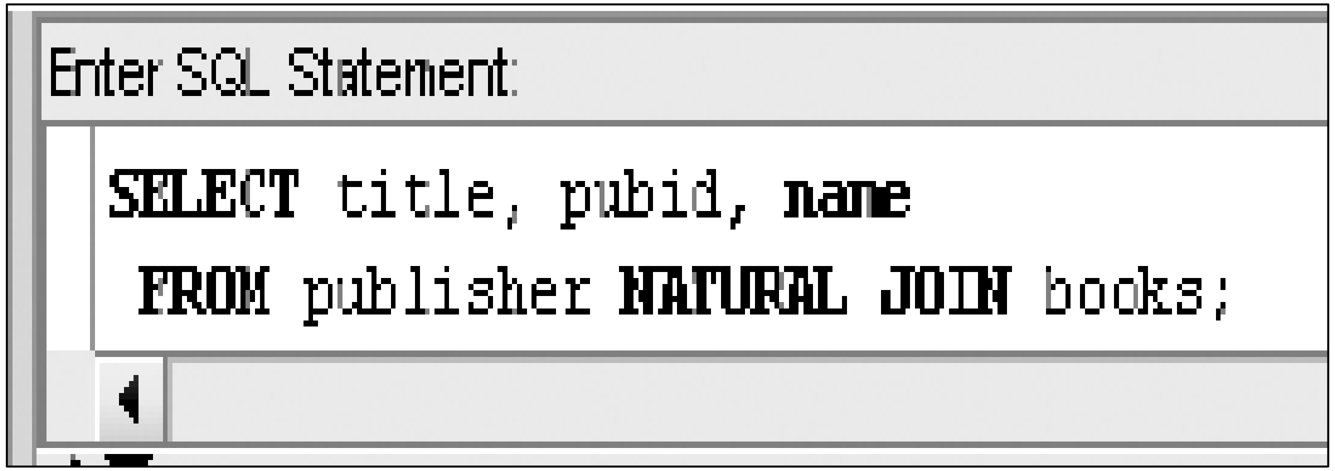


```
SELECT c.lastname, c.firstname, b.title
 FROM customers c, orders o, orderitems oi, books b
 WHERE c.customer# = o.customer#
   AND o.order# = oi.order#
   AND oi.isbn = b.isbn
 ORDER BY lastname, firstname;
```

KING GRADUATE SCHOOL MONROE COLLEGE

Equality Joins: NATURAL JOIN…

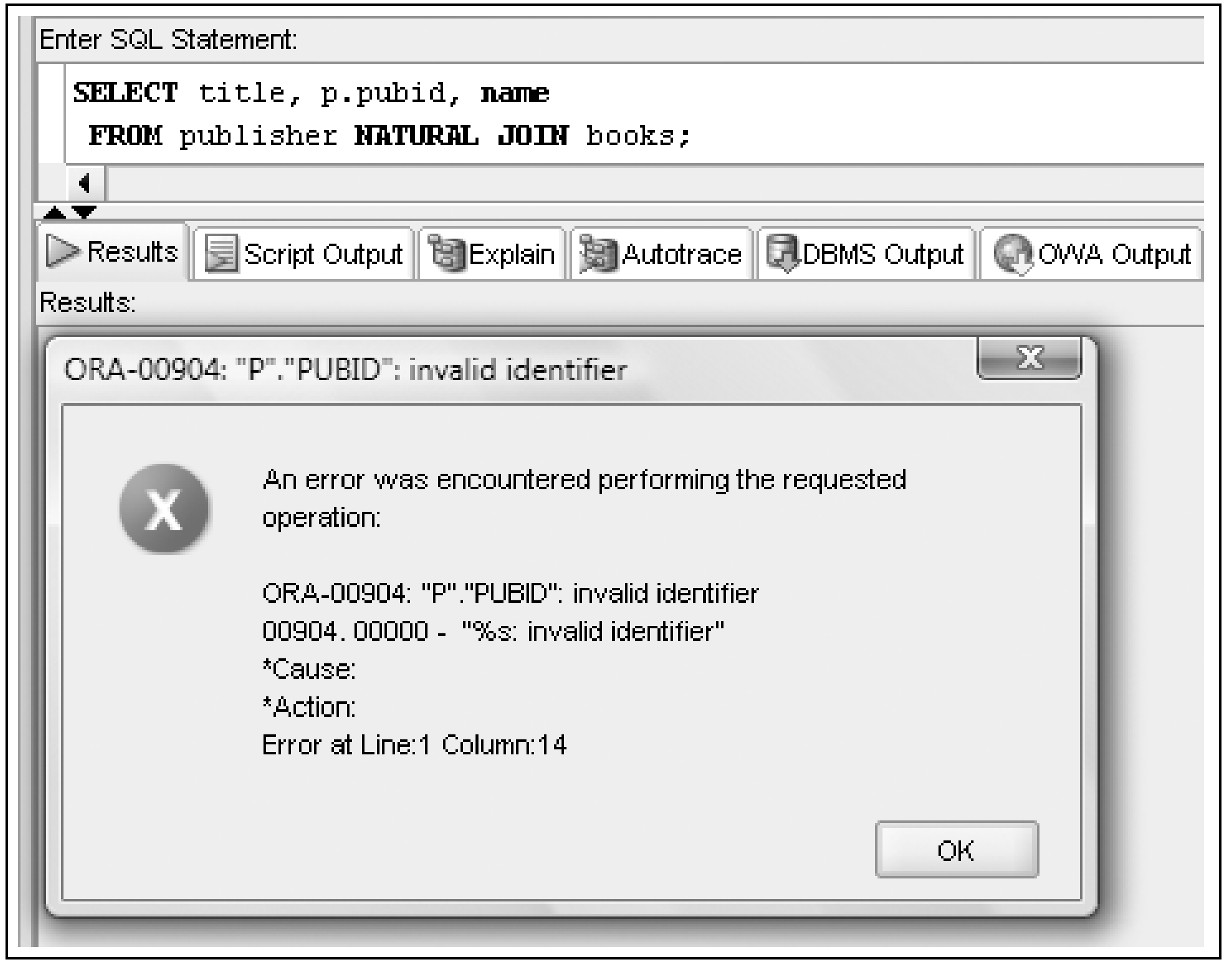
- Equality Joins Perform a JOIN Against Equality or Matching Columns…
- NATURAL JOIN Creates a Join Automatically Between Two Tables Based on Columns With Matching Names…
- It Implies That the Tables in the FROM Clause Have at Least One Common Column…
- You Can NOT Put a Qualifier on the Common Column…



```
SELECT title, pubid, name
 FROM publisher NATURAL JOIN books;
```

No Qualifiers with a NATURAL JOIN…

- Notice the Qualifier in the SELECT Clause Causes an Error…
- It Makes No Logical Sense to Qualify a Column That is Supposed to Be the Same in Both Tables…



Enter SQL Statement:

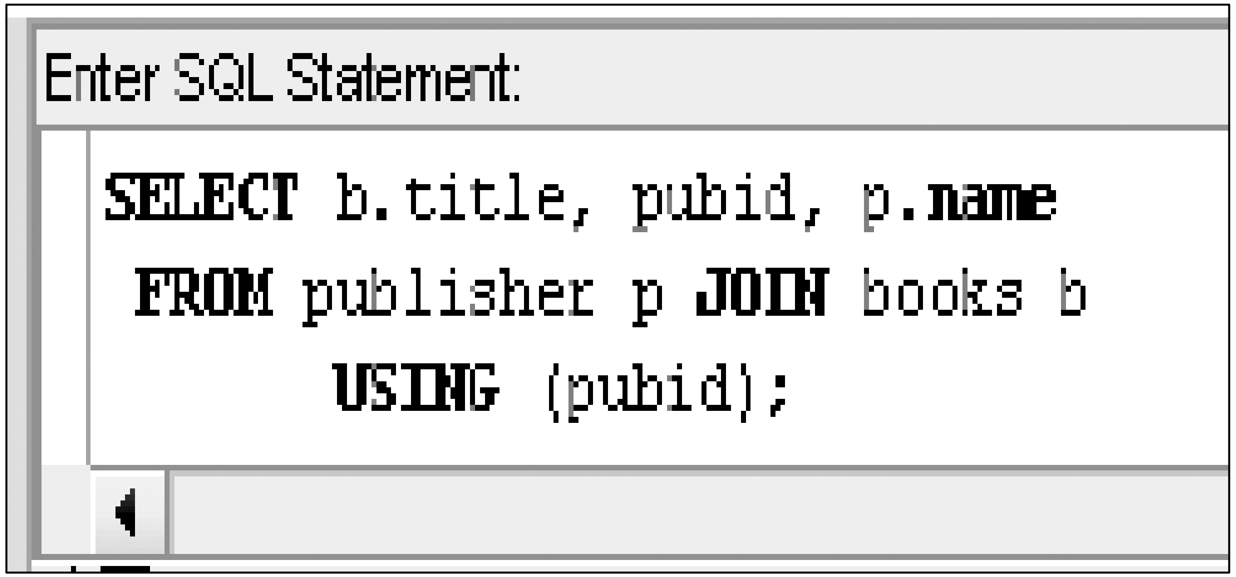
```
SELECT title, p.pubid, name
 FROM publisher NATURAL JOIN books;
```

```
ORA-00904: "P"."PUBID": invalid identifier

An error was encountered performing the requested
operation:

ORA-00904: "P"."PUBID": invalid identifier
00904. 00000 -  "%s: invalid identifier"
*Cause:
*Action:
Error at Line:1 Column:14
```

KING GRADUATE SCHOOL MONROE COLLEGE

Equality Joins: JOIN…USING…

- Allow You to Create Joins Based on a Column That Has the Same ***Name*** and Definition in Both Tables…
- The USING Clause Defines the Common Field…
- JOIN…USING is a Better Option Than a NATURAL JOIN…It Forces You to Verify the Relationships…

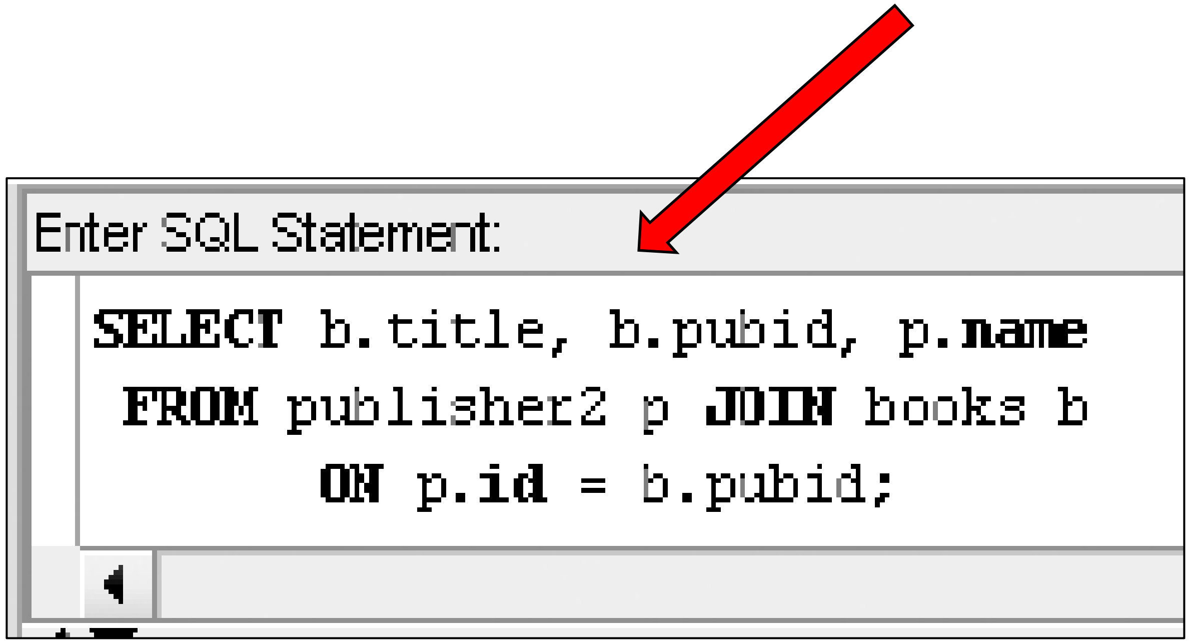


Enter SQL Statement:

```
SELECT b.title, pubid, p.name
 FROM publisher p JOIN books b
       USING (pubid);
```

Equality Joins: JOIN…ON…

- Used to JOIN Tables on a "Common" Field When the Field Name in Each Table is Different…
- Notice in This Case the Field Names Are Qualified…


Enter SQL Statement:

```
SELECT b.title, b.pubid, p.name
 FROM publisher2 p JOIN books b
       ON p.id = b.pubid;
```

KING GRADUATE SCHOOL MONROE COLLEGE

JOIN Keyword Overview…

- Use JOIN…USING When Tables Have One or More Columns in Common…
- Use JOIN…ON When Same Named Columns Are Not Involved or a Condition is Needed to Specify a Relationship Other Than Equivalency (Next Section)…
- Using the JOIN Keyword Frees the WHERE Clause For Exclusive Use in Restricting Rows…

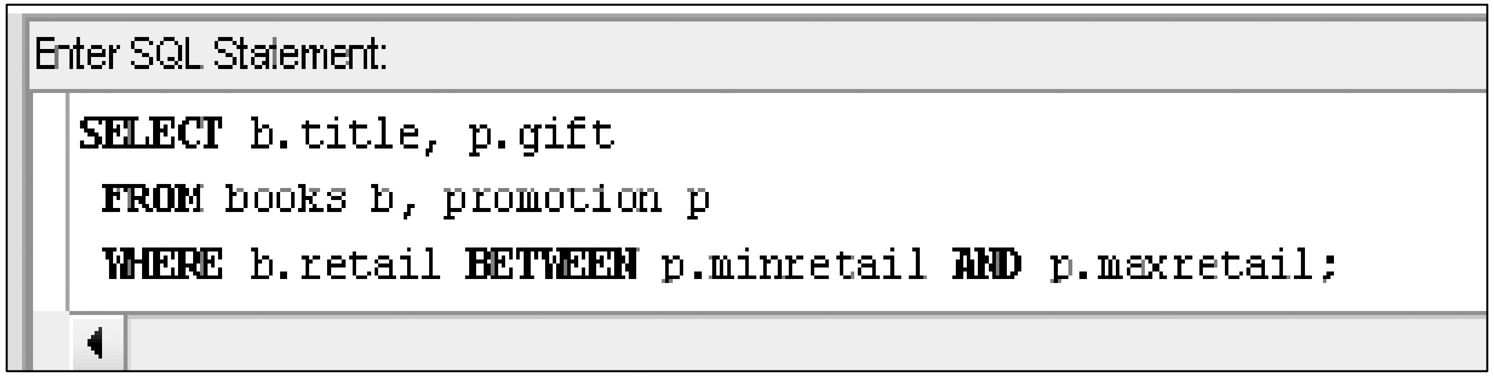
Non-Equality JOINS – Part 1…

- A JOIN Where the Match Column Values From Different Tables Are Based on an Inequality…
- For Example, >, <, >=, <= …
- The Value of the Join Column in Each Row in the Source Table is Compared to the Corresponding Values in the Target Table…
- A Match is Found If the Expression Based on the Inequality Operator Used in the JOIN Evaluates to True…

KING GRADUATE SCHOOL MONROE COLLEGE

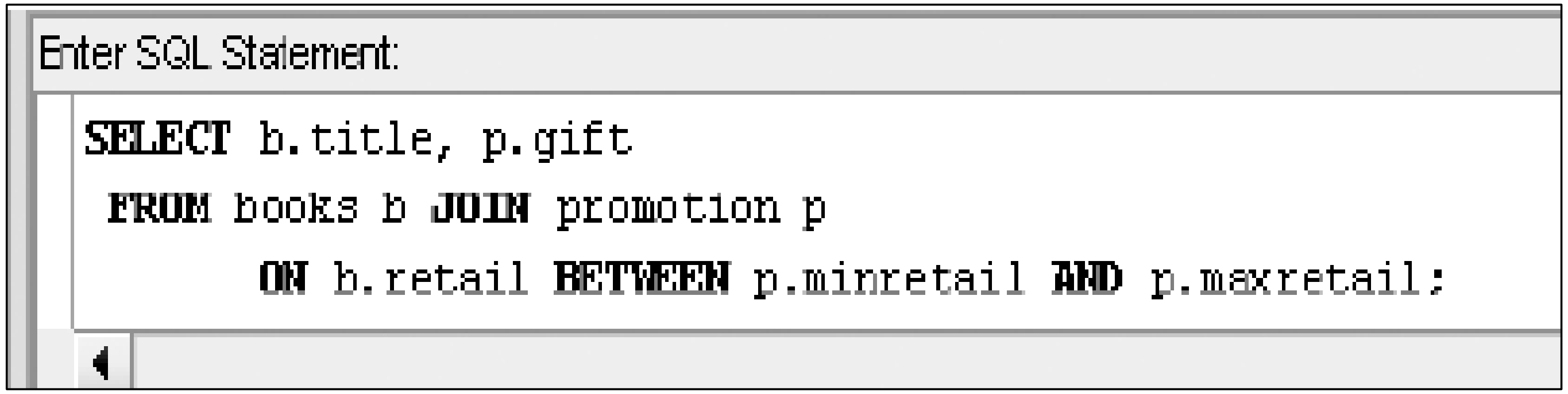
Non-Equality JOINS – Part 2…

- In WHERE Clause, Use Any Comparison Operator Other Than the Equal Sign…



```
SELECT b.title, p.gift
 FROM books b, promotion p
 WHERE b.retail BETWEEN p.minretail AND p.maxretail;
```

- In FROM Clause, Use JOIN…ON Keywords With a Non-Equivalent Condition…



```
SELECT b.title, p.gift
 FROM books b JOIN promotion p
       ON b.retail BETWEEN p.minretail AND p.maxretail;
```

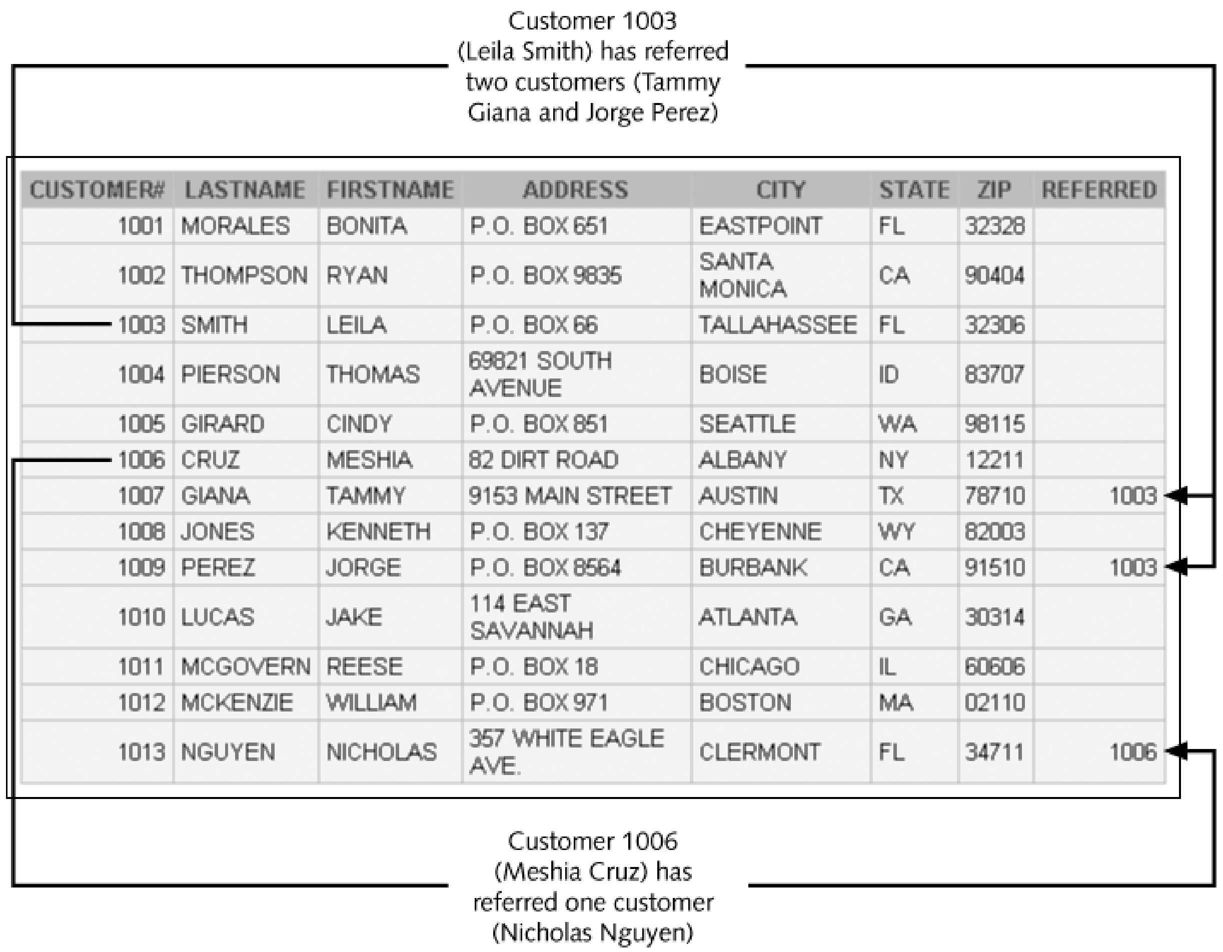
KING GRADUATE SCHOOL MONROE COLLEGE

Self-JOINS…

- Used to Link a Table to Itself…
- Requires the Use of Table Aliases…
- Requires the Use of a Column Qualifier…
- WHAT?!...

Customer Table Example – Part 1…

- For Example…
- Customers Who Refer a New Customer to JustLee Books Receive a Discount Certificate for a Future Purchase…
- We Want the Name of the Person Who Referred the Customer…



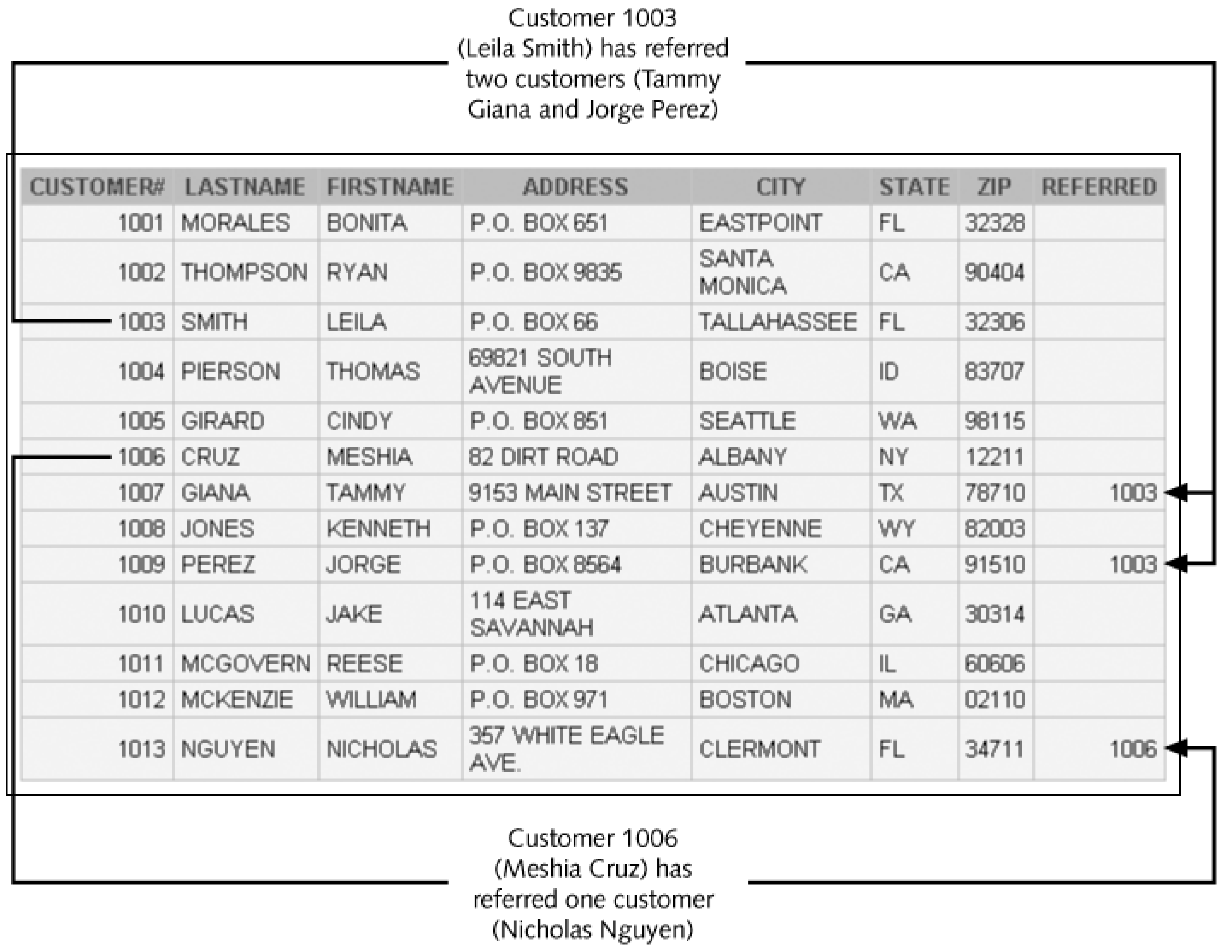
Customer 1003 (Leila Smith) has referred two customers (Tammy Giana and Jorge Perez)

CUSTOMER#	LASTNAME	FIRSTNAME	ADDRESS	CITY	STATE	ZIP	REFERRED
1001	MORALES	BONITA	P.O. BOX 651	EASTPOINT	FL	32328	
1002	THOMPSON	RYAN	P.O. BOX 9835	SANTA MONICA	CA	90404	
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	
1006	CRUZ	MESHIA	82 DIRT ROAD	ALBANY	NY	12211	
1007	GIANA	TAMMY	9153 MAIN STREET	AUSTIN	TX	78710	1003
1008	JONES	KENNETH	P.O. BOX 137	CHEYENNE	WY	82003	
1009	PEREZ	JORGE	P.O. BOX 8564	BURBANK	CA	91510	1003
1010	LUCAS	JAKE	114 EAST SAVANNAH	ATLANTA	GA	30314	
1011	MCGOVERN	REESE	P.O. BOX 18	CHICAGO	IL	60606	
1012	MCKENZIE	WILLIAM	P.O. BOX 971	BOSTON	MA	02110	
1013	NGUYEN	NICHOLAS	357 WHITE EAGLE AVE.	CLERMONT	FL	34711	1006

Customer 1006 (Meshia Cruz) has referred one customer (Nicholas Nguyen)

Customer Table Example – Part 2…

- The REFERRED Column of the CUSTOMERS Table Stores the Customer Number of the Person Who Referred the New Customer…
- The CUSTOMERS Table Also Has the Name of the Person Who Referred the New Customer…
- To Retrieve the Information You Need to JOIN the Table to Itself…



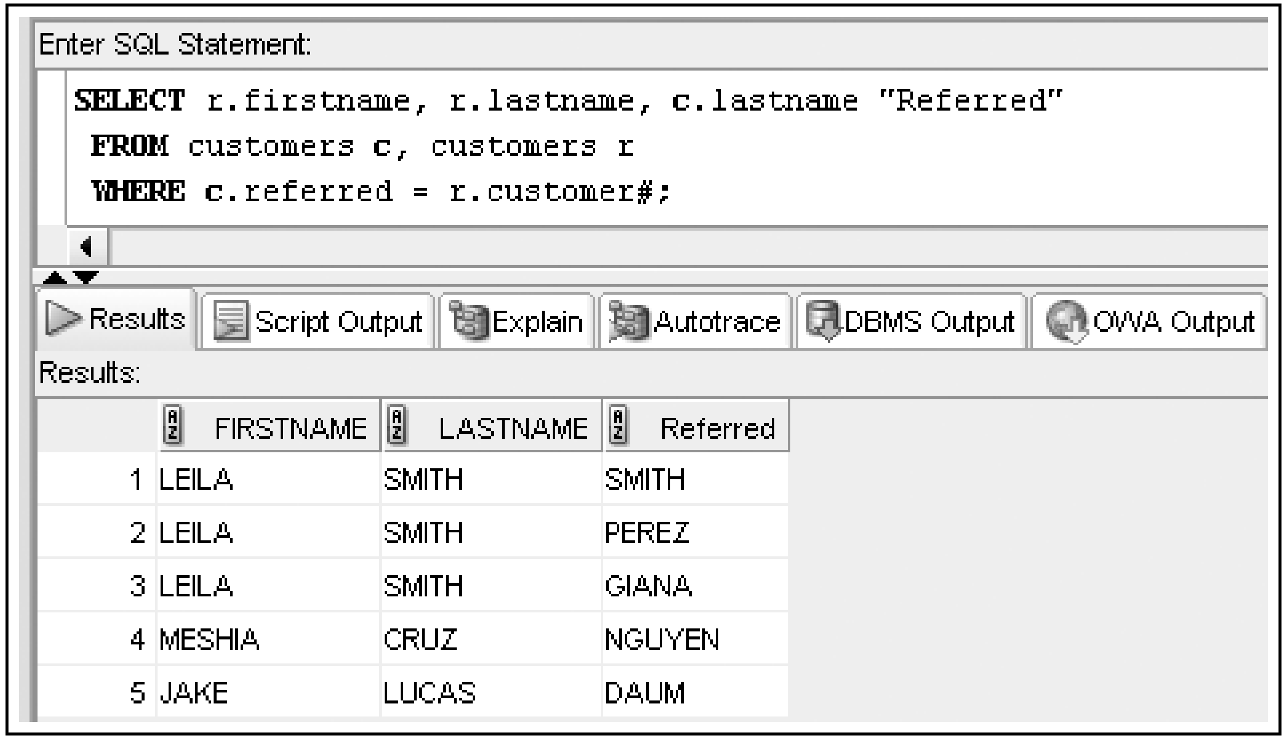
Customer 1003 (Leila Smith) has referred two customers (Tammy Giana and Jorge Perez)

CUSTOMER#	LASTNAME	FIRSTNAME	ADDRESS	CITY	STATE	ZIP	REFERRED
1001	MORALES	BONITA	P.O. BOX 651	EASTPOINT	FL	32328	
1002	THOMPSON	RYAN	P.O. BOX 9835	SANTA MONICA	CA	90404	
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	
1006	CRUZ	MESHIA	82 DIRT ROAD	ALBANY	NY	12211	
1007	GIANA	TAMMY	9153 MAIN STREET	AUSTIN	TX	78710	1003
1008	JONES	KENNETH	P.O. BOX 137	CHEYENNE	WY	82003	
1009	PEREZ	JORGE	P.O. BOX 8564	BURBANK	CA	91510	1003
1010	LUCAS	JAKE	114 EAST SAVANNAH	ATLANTA	GA	30314	
1011	MCGOVERN	REESE	P.O. BOX 18	CHICAGO	IL	60606	
1012	MCKENZIE	WILLIAM	P.O. BOX 971	BOSTON	MA	02110	
1013	NGUYEN	NICHOLAS	357 WHITE EAGLE AVE.	CLERMONT	FL	34711	1006

Customer 1006 (Meshia Cruz) has referred one customer (Nicholas Nguyen)

Self-Joins: WHERE Clause Example…

- Data in Once Column of a Table Has a Relationship With Another Column in the Same Table…
- You Must Make It Appear as If the Data is Coming From Two Tables…
- To Do That You Need to Use an Alias…



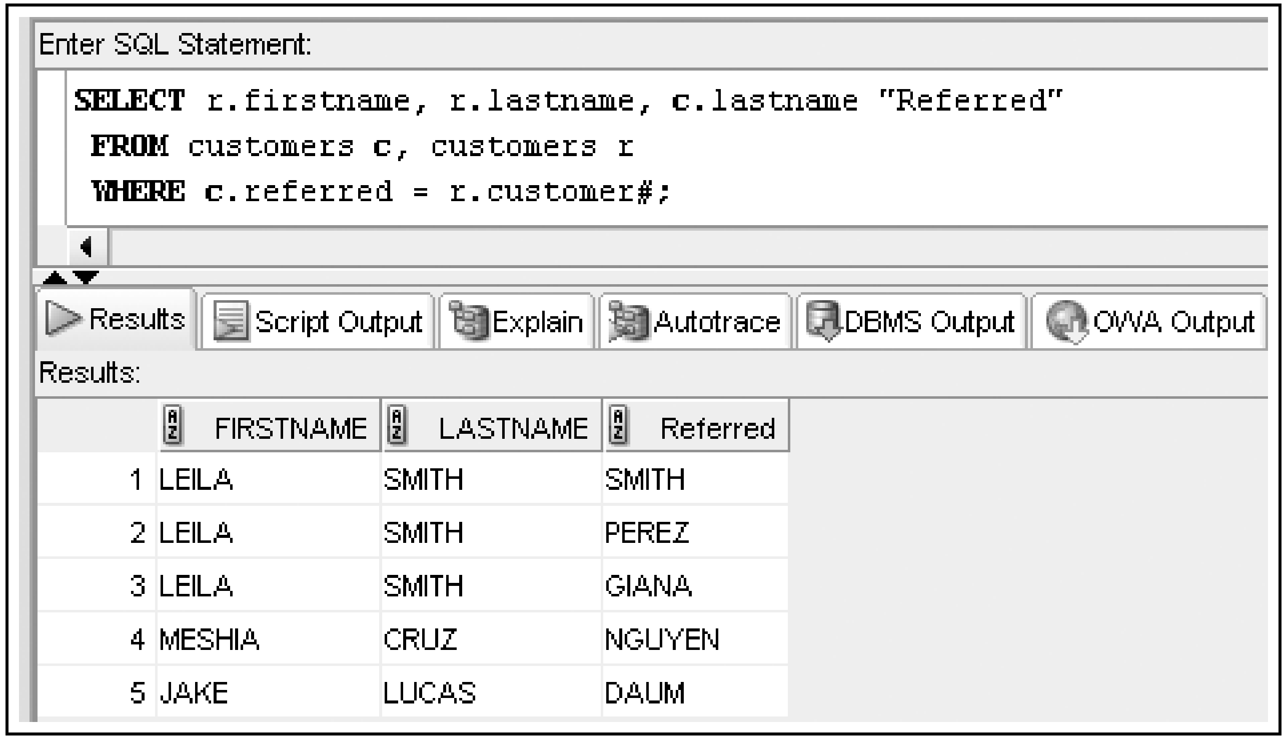
```
SELECT r.firstname, r.lastname, c.lastname "Referred"
 FROM customers c, customers r
 WHERE c.referred = r.customer#;
```

	FIRSTNAME	LASTNAME	Referred
1	LEILA	SMITH	SMITH
2	LEILA	SMITH	PEREZ
3	LEILA	SMITH	GIANA
4	MESHIA	CRUZ	NGUYEN
5	JAKE	LUCAS	DAUM

KING GRADUATE SCHOOL MONROE COLLEGE

Self-Joins: WHERE Clause Example…

- "c" Refers to the Customer Who Referred the New Customer…
- "r" Refers to the Customer They Referred…
- Oracle Treats Each Alias as If It Were a Different Table…
- Using the WHERE Clause is Considered the "Traditional" Method of Doing This…

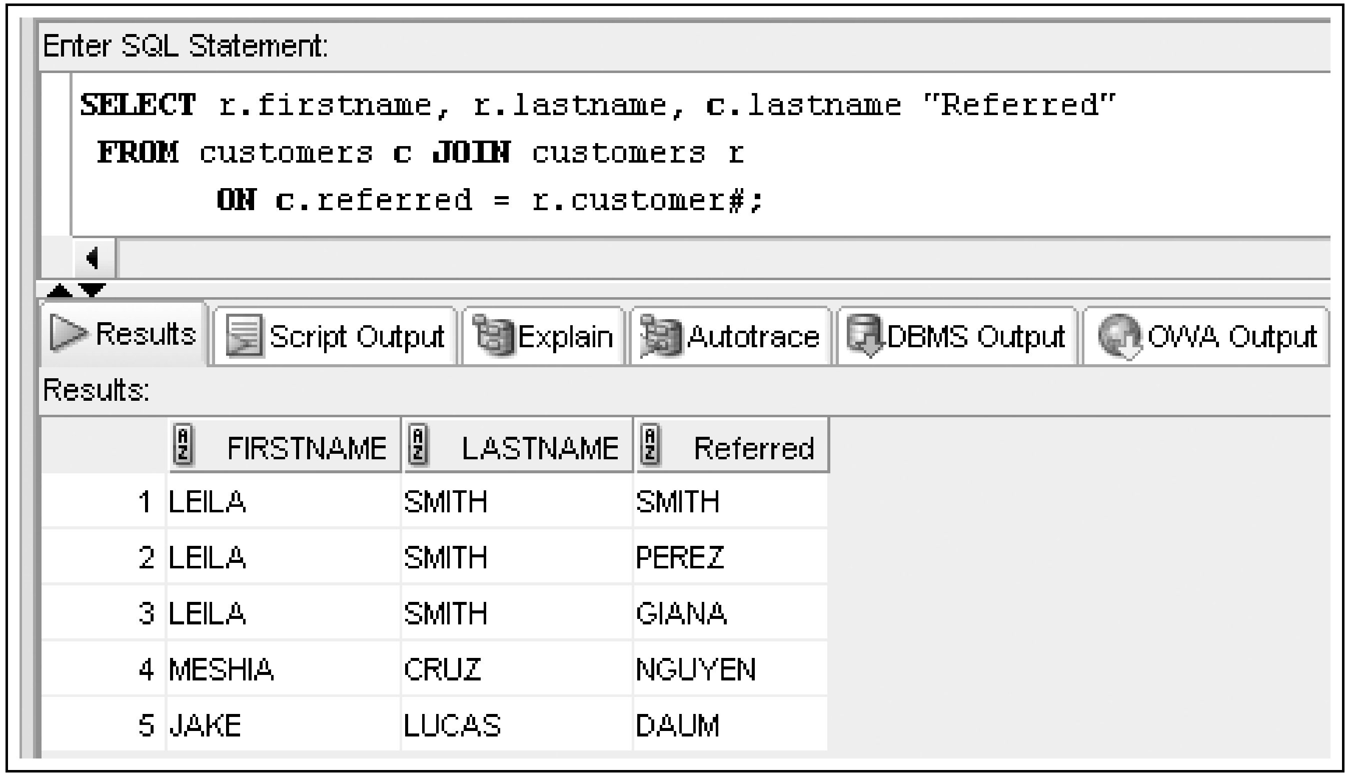


```
SELECT r.firstname, r.lastname, c.lastname "Referred"
 FROM customers c, customers r
 WHERE c.referred = r.customer#;
```

	FIRSTNAME	LASTNAME	Referred
1	LEILA	SMITH	SMITH
2	LEILA	SMITH	PEREZ
3	LEILA	SMITH	GIANA
4	MESHIA	CRUZ	NGUYEN
5	JAKE	LUCAS	DAUM

Self-Joins: JOIN…ON Example…

- This Produces the Same Results as the Prior SQL…
- NOTE…a USING Clause Can NOT Be Used Here Because Two Different Column Names Are Used in the JOIN
  - ✓"customers ***c***"…
  - ✓"customers ***r***"…



Enter SQL Statement:

```
SELECT r.firstname, r.lastname, c.lastname "Referred"
 FROM customers c JOIN customers r
       ON c.referred = r.customer#;
```

Results:

	FIRSTNAME	LASTNAME	Referred
1	LEILA	SMITH	SMITH
2	LEILA	SMITH	PEREZ
3	LEILA	SMITH	GIANA
4	MESHIA	CRUZ	NGUYEN
5	JAKE	LUCAS	DAUM

KING GRADUATE SCHOOL MONROE COLLEGE

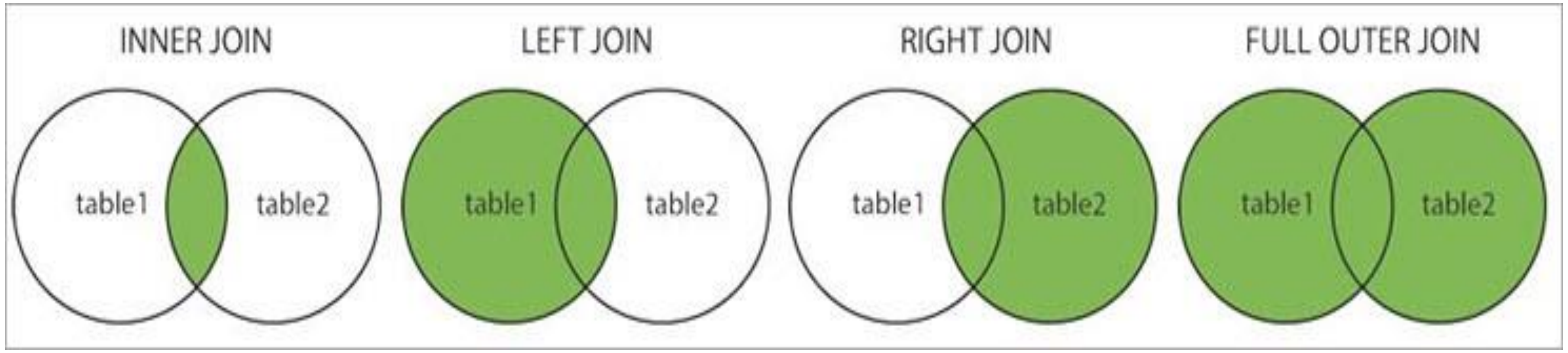
Inner Vs. Outer JOIN…

- INNER JOIN Returns Only the Rows That Have Matching Values in BOTH Tables…All the JOINS Done So Far Are Technically INNER JOINS…
- OUTER JOIN Includes Matching and Some Non-Matching Rows Between the Tables…
- OUTER JOINs Differs From INNER JOINs in How They Handle the False Match Condition…

KING GRADUATE SCHOOL MONROE COLLEGE

Types of OUTER JOINs…

- **LEFT OUTER JOIN**: Returns All Rows From the Left Table and Matching Records Between Both the Tables…
- **RIGHT OUTER JOIN**: Returns All Rows From the Right Table and Matching Records Between Both the Tables…
- **FULL OUTER JOIN**: Combines the Result of the Left Outer Join and Right Outer Join…



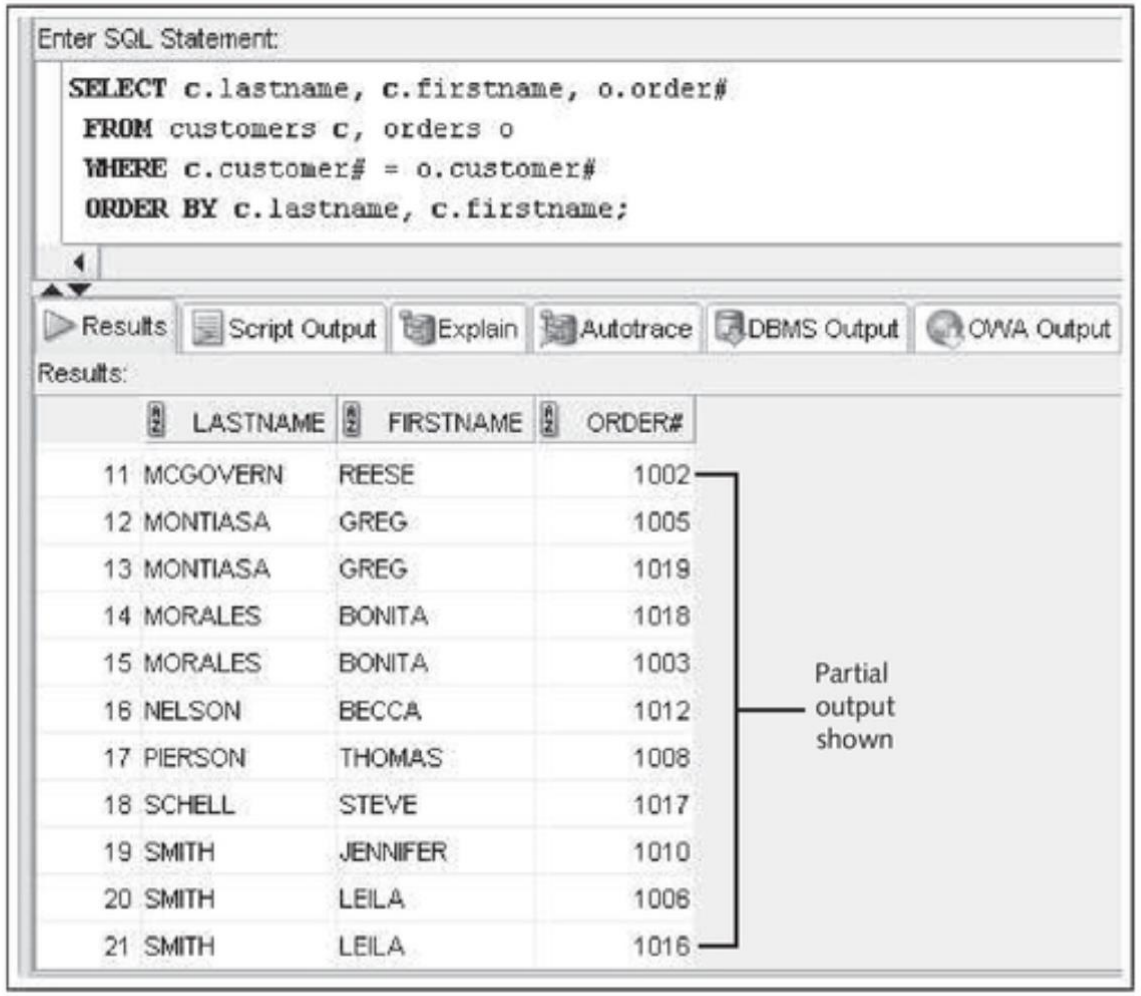
KING GRADUATE SCHOOL MONROE COLLEGE

Outer Joins…

- Use Outer Joins to Include Rows That Do Not Have a Match in the Other Table…
- In WHERE Clause, Include Outer Join Operator (+) Immediately After the Column Name of the Table With Missing Rows to Add NULL Rows…
- In FROM Clause, Use FULL, LEFT, Or RIGHT With OUTER JOIN Keywords…

INNER JOIN…

- What is This Query Doing?...
- This Query Identifies Any Customer Who Has Place an Order Stored in the ORDERS Table…
- It Does NOT List Customers Who Have Not Placed an Order…
- The INNER JOIN Omit Rows That Do Not Match…


Enter SQL Statement:

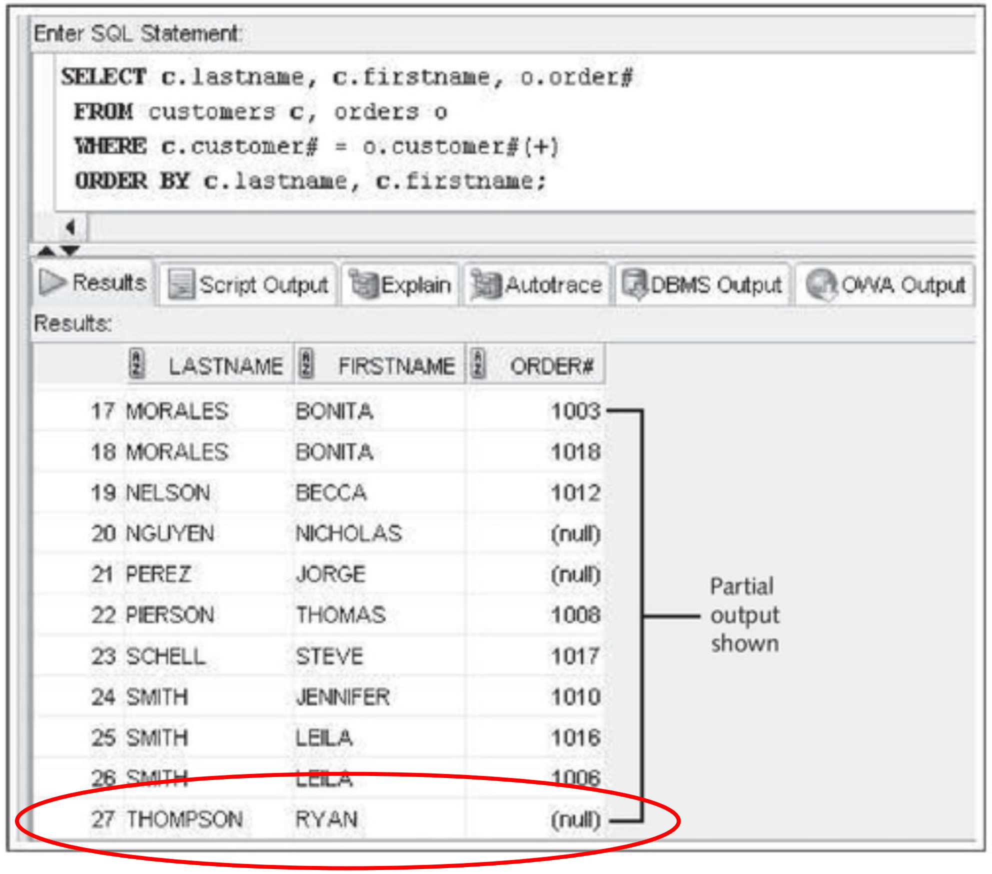
```
SELECT c.lastname, c.firstname, o.order#
 FROM customers c, orders o
 WHERE c.customer# = o.customer#
 ORDER BY c.lastname, c.firstname;
```

	LASTNAME	FIRSTNAME	ORDER#
11	MCGOVERN	REESE	1002
12	MONTIASA	GREG	1005
13	MONTIASA	GREG	1019
14	MORALES	BONITA	1018
15	MORALES	BONITA	1003
16	NELSON	BECCA	1012
17	PIERSON	THOMAS	1008
18	SCHELL	STEVE	1017
19	SMITH	JENNIFER	1010
20	SMITH	LEILA	1006
21	SMITH	LEILA	1016

KING GRADUATE SCHOOL MONROE COLLEGE

A “Traditional” OUTER JOIN…

- To Create a Row for Records With No Matching Row (NULL Row), Use the Outer Join Operator (+)…
- It is Placed in the WHERE Clause Immediately After the Column Name From the Table Without a Matching Row…The Field is Marked as “NULL”…
- The ORDERS Table is Referred to as the “Deficient Table” Because It is Missing Data…
- Two Notes:
  - ✓ (+) Can Only Be Used on ONE Table…You Can Not Create NULL Rows in Both Tables…
  - ✓ (+) Can NOT Be Used in a Condition That Uses the IN or OR Operator…

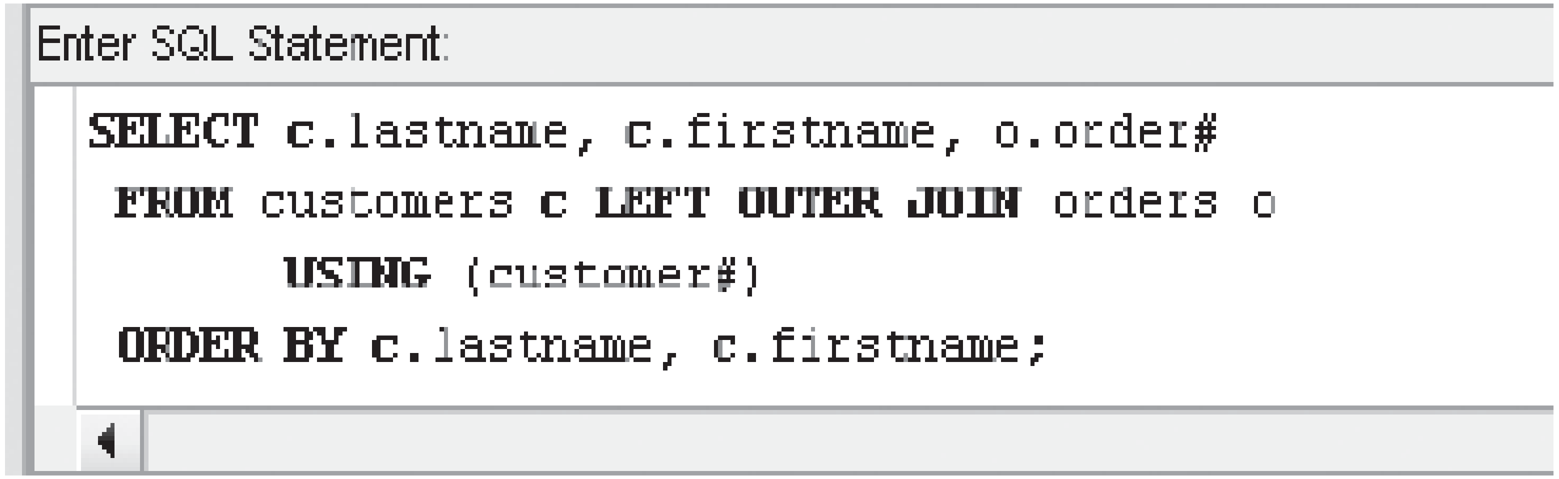


```
SELECT c.lastname, c.firstname, o.order#
 FROM customers c, orders o
 WHERE c.customer# = o.customer#(+)
 ORDER BY c.lastname, c.firstname;
```

```
    LASTNAME    FIRSTNAME   ORDER#
 17 MORALES     BONITA        1003
 18 MORALES     BONITA        1018
 19 NELSON      BECCA         1012
 20 NGUYEN      NICHOLAS    (null)
 21 PEREZ       JORGE       (null)
 22 PIERSON     THOMAS        1008
 23 SCHELL      STEVE         1017
 24 SMITH       JENNIFER      1010
 25 SMITH       LEILA         1016
 26 SMITH       LEILA         1006
 27 THOMPSON    RYAN        (null)
```

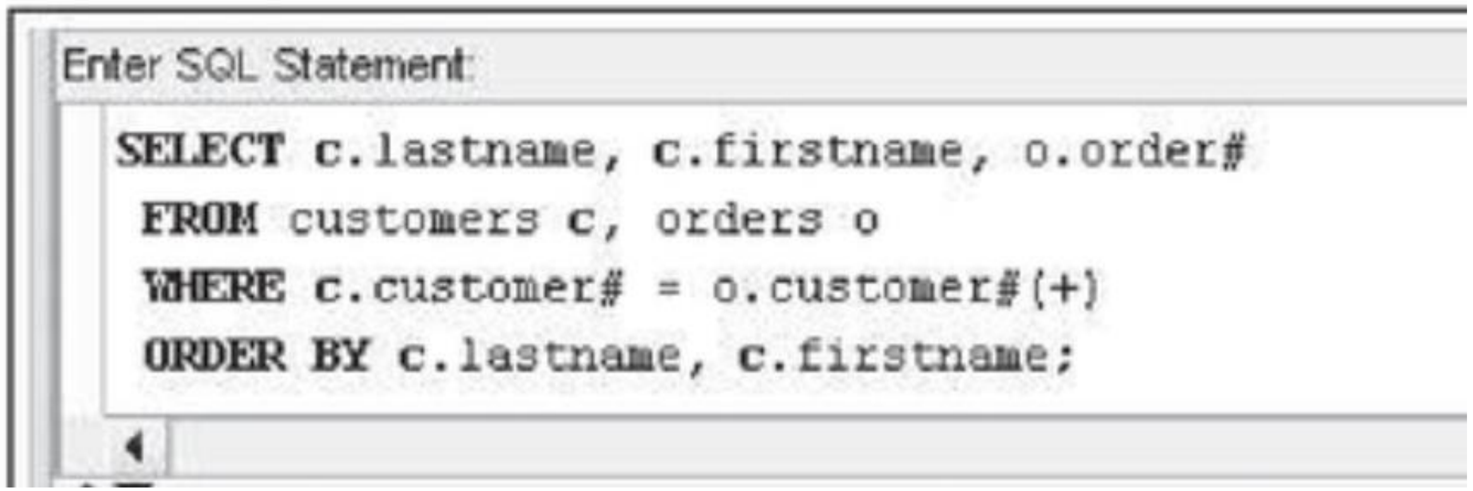
OUTER JOINs…

- This SQL Does the Same as the Prior SQL But Uses the LEFT OUTER JOIN Keyword…
- The JOIN Keyword Allows You to Specify Which Table the JOIN Should Apply To (LEFT, RIGHT, or FULL)…
- LEFT, RIGHT, or FULL is Based On the Table’s Location in the JOIN Condition…
- For Example, a LEFT OUTER JOIN Keeps All Rows in the Table Listed On the Left Side Of the Join Condition, Even If No Matches Are Found With The Table Listed On The Right….a FULL OUTER JOIN Keeps All Rows From Both Tables In The Results, No Matter Which Table Is Deficient When Matching Rows…

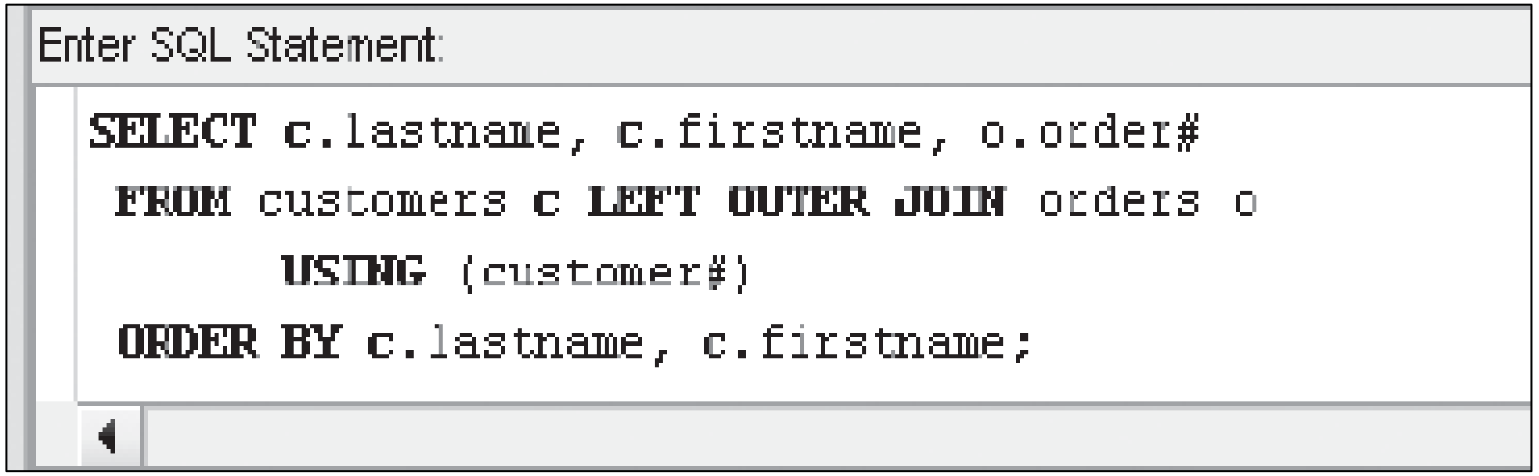


```
SELECT c.lastname, c.firstname, o.order#
 FROM customers c LEFT OUTER JOIN orders o
       USING (customer#)
 ORDER BY c.lastname, c.firstname;
```

Side by Side…


Enter SQL Statement:

```
SELECT c.lastname, c.firstname, o.order#
 FROM customers c, orders o
 WHERE c.customer# = o.customer#(+)
 ORDER BY c.lastname, c.firstname;
```

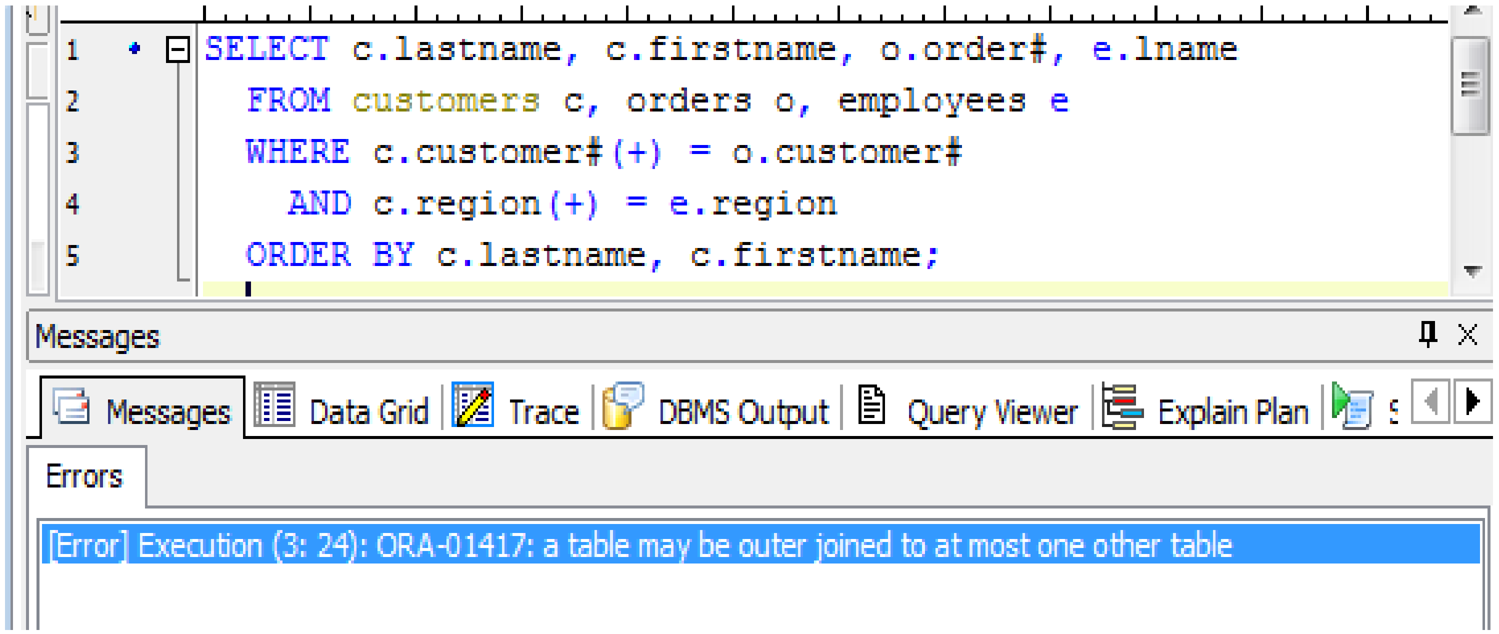

Enter SQL Statement:

```
SELECT c.lastname, c.firstname, o.order#
 FROM customers c LEFT OUTER JOIN orders o
       USING (customer#)
 ORDER BY c.lastname, c.firstname;
```

KING GRADUATE SCHOOL MONROE COLLEGE

Outer Joins – Part 2…

- If Multiple Join Conditions Are Used, the Outer Join Condition May Be Required in All of the Join Conditions to Retain Nonmatching Rows…
- In Previous Versions of Oracle an Error Would Be Raised if the Outer Join Operator is Used on the Same Table in More Than One Join Operation…



```
SELECT c.lastname, c.firstname, o.order#, e.lname
  FROM customers c, orders o, employees e
  WHERE c.customer#(+) = o.customer#
    AND c.region(+) = e.region
  ORDER BY c.lastname, c.firstname;
```

[Error] Execution (3: 24): ORA-01417: a table may be outer joined to at most one other table

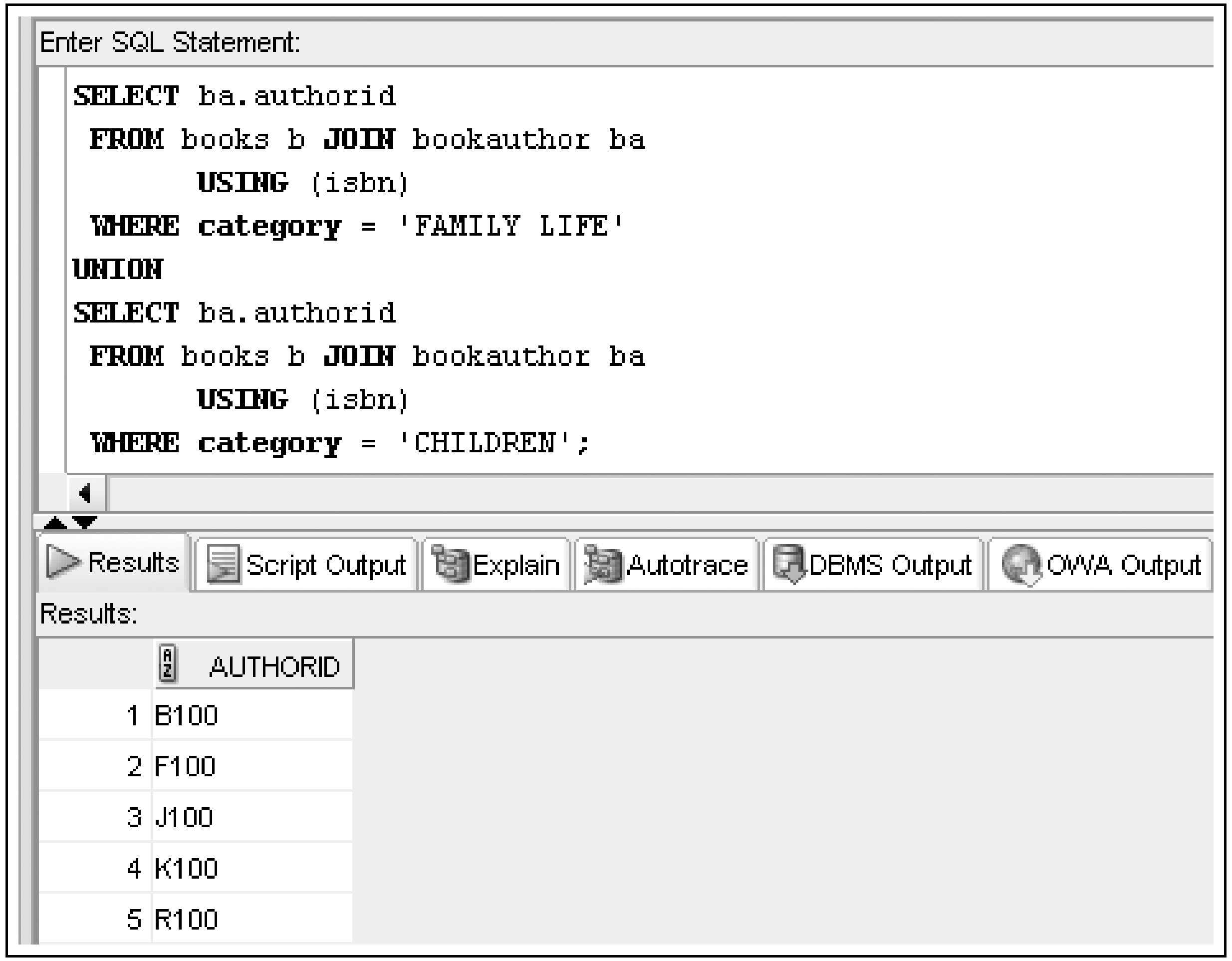
KING GRADUATE SCHOOL MONROE COLLEGE

Set Operators…

- Used to Combine the Results of Two or More SELECT Statements…
- **UNION :** Returns the Results of BOTH Queries and REMOVES Duplicates…
- **UNION ALL :** Returns the Results of BOTH Queries But INCLUDES Duplicates…
- **INTERSECT :** Returns Only the Rows Included in the Results of BOTH Queries…
- **MINUS :** Subtracts the Second Query's Results If They Are Returned in the First Query's Results…

Set Operators : UNION Example…

- **UNION :** Returns the Results of BOTH Queries and REMOVES Duplicates…



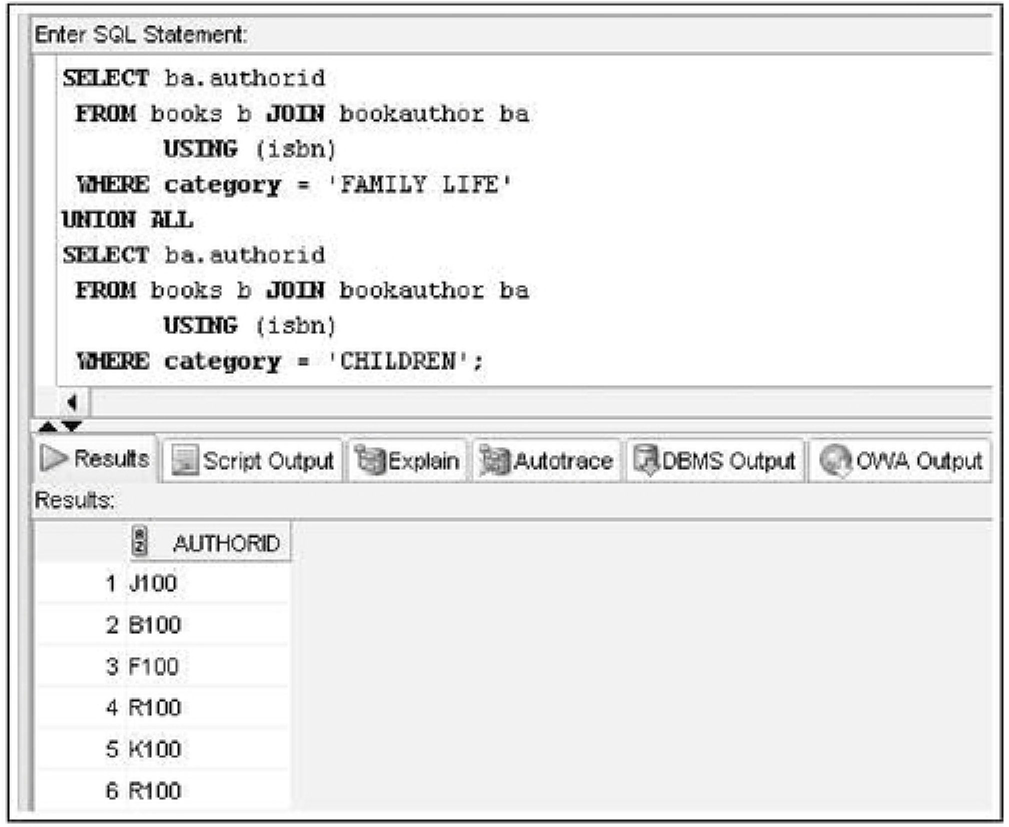
```
SELECT ba.authorid
 FROM books b JOIN bookauthor ba
       USING (isbn)
 WHERE category = 'FAMILY LIFE'
UNION
SELECT ba.authorid
 FROM books b JOIN bookauthor ba
       USING (isbn)
 WHERE category = 'CHILDREN';
```

	AUTHORID
1	B100
2	F100
3	J100
4	K100
5	R100

KING GRADUATE SCHOOL MONROE COLLEGE

Set Operators : UNION ALL Example…

- **UNION ALL :** Returns the Results of BOTH Queries But INCLUDES Duplicates…



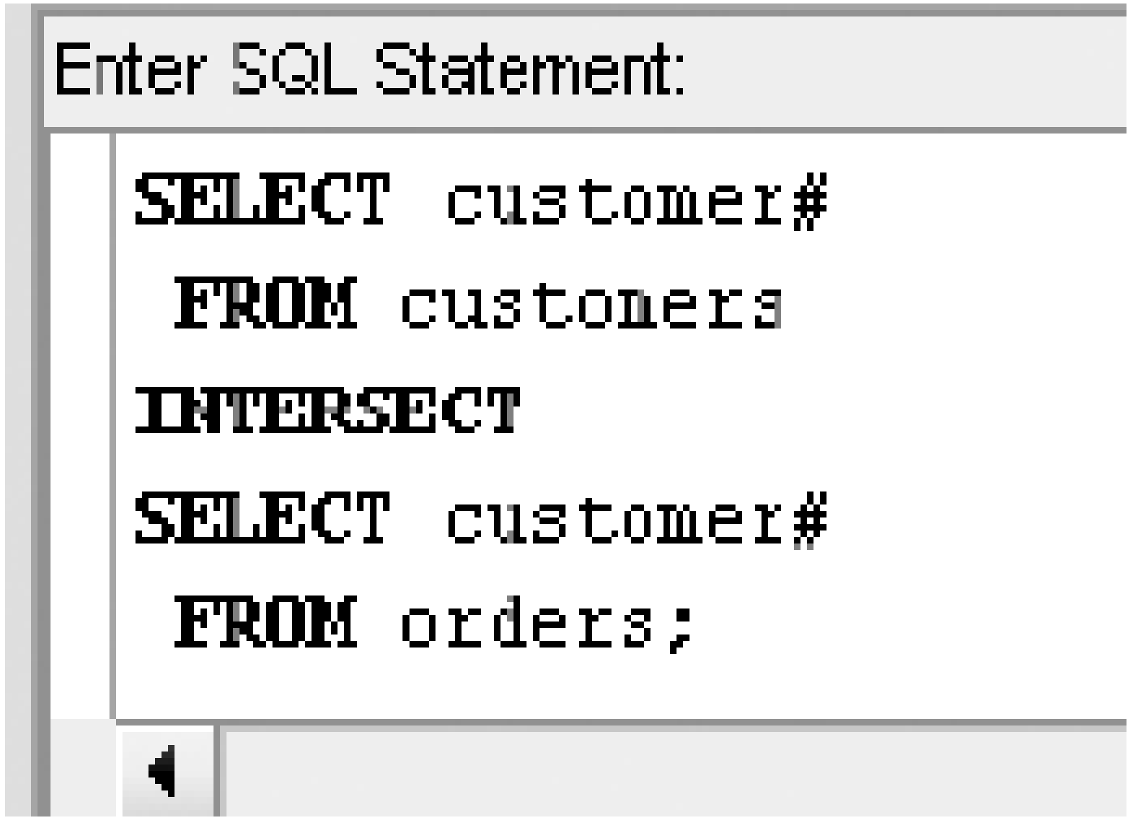
```
SELECT ba.authorid
 FROM books b JOIN bookauthor ba
       USING (isbn)
 WHERE category = 'FAMILY LIFE'
UNION ALL
SELECT ba.authorid
 FROM books b JOIN bookauthor ba
       USING (isbn)
 WHERE category = 'CHILDREN';
```

```
  AUTHORID
1 J100
2 B100
3 F100
4 R100
5 K100
6 R100
```

KING GRADUATE SCHOOL MONROE COLLEGE

Set Operators : INTERSECT Example…

- **INTERSECT :** Returns Only the Rows Included in the Results of BOTH Queries…

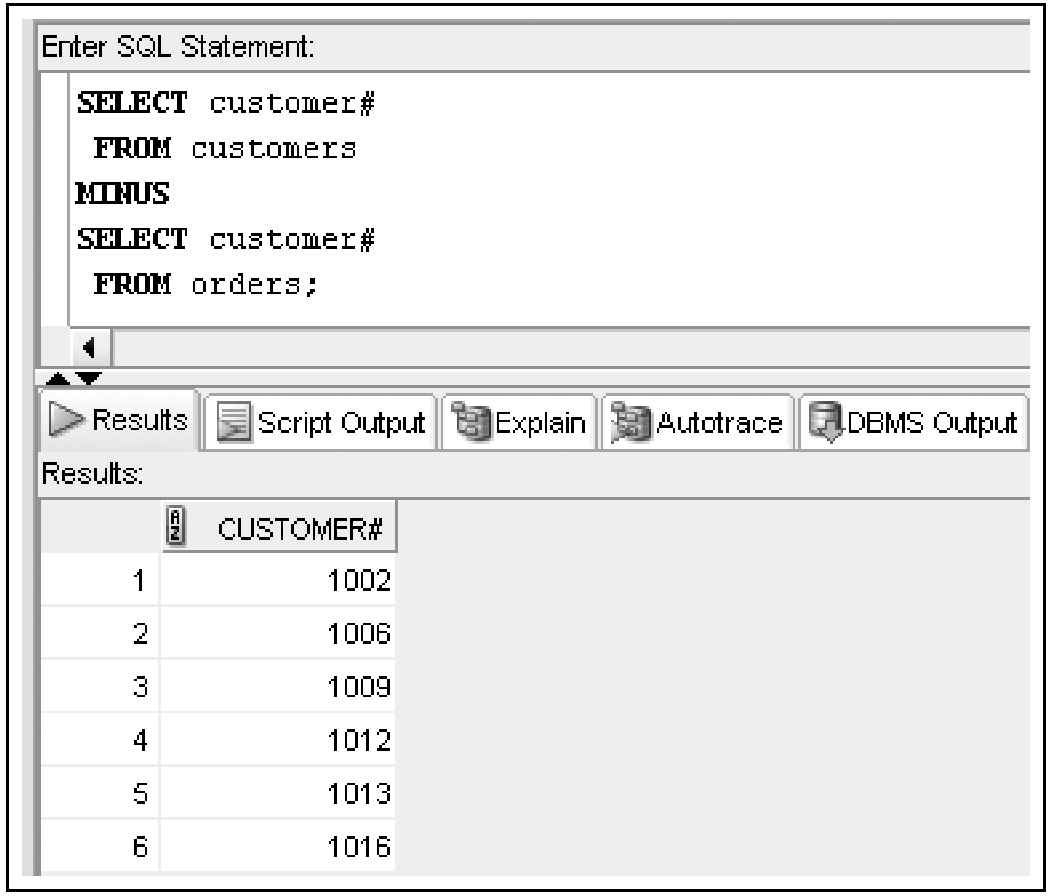

Enter SQL Statement:

```
SELECT customer#
 FROM customers
INTERSECT
SELECT customer#
 FROM orders;
```

KING GRADUATE SCHOOL MONROE COLLEGE

Set Operators : MINUS Example…

- **MINUS :** Subtracts the Second Query's Results If They Are Returned in the First Query's Results…



```
SELECT customer#
 FROM customers
MINUS
SELECT customer#
 FROM orders;
```

	CUSTOMER#
1	1002
2	1006
3	1009
4	1012
5	1013
6	1016

Questions…