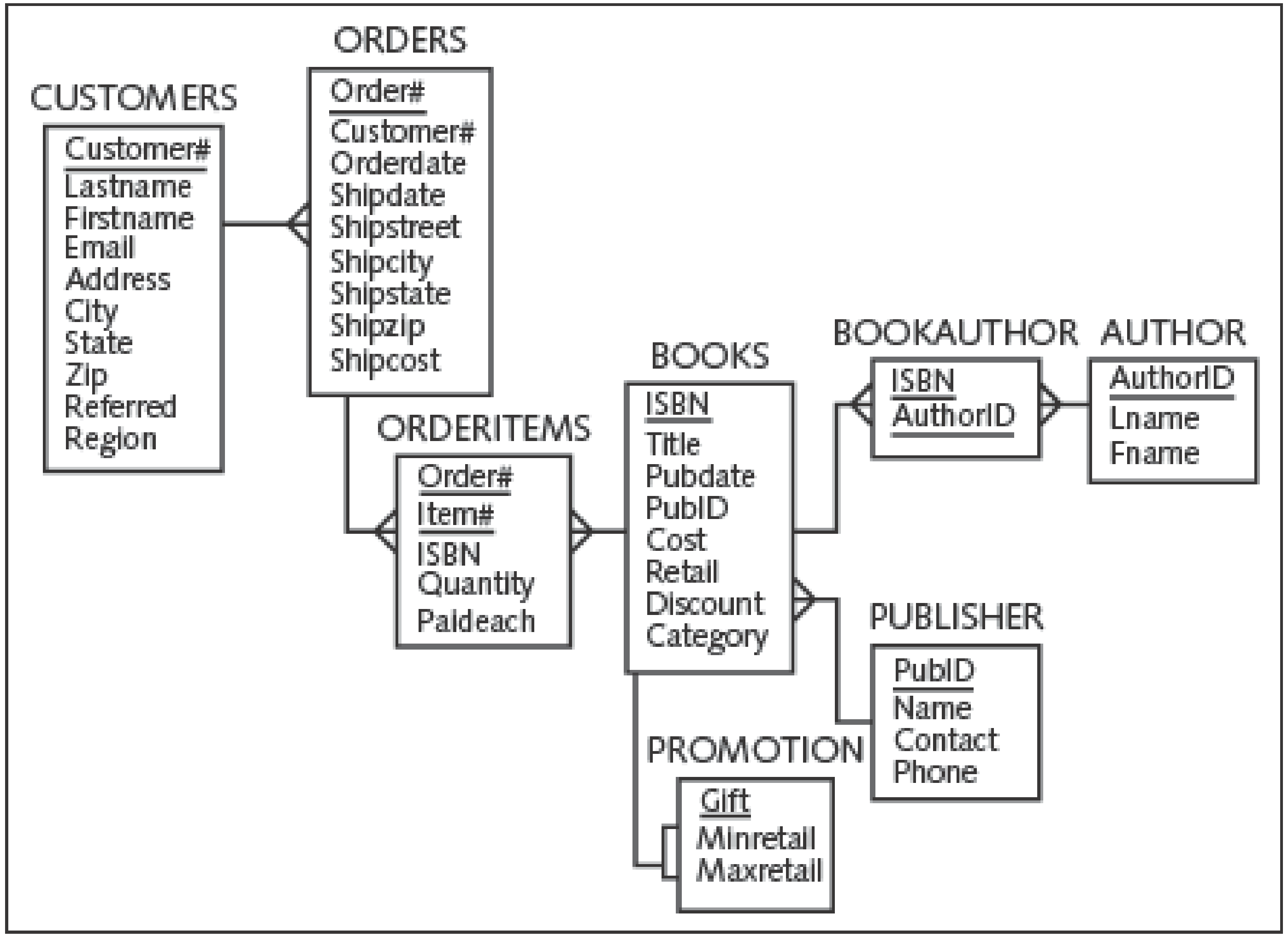
Database Systems Subqueries and Merges

CS 630 Database Systems

Professor Nardi

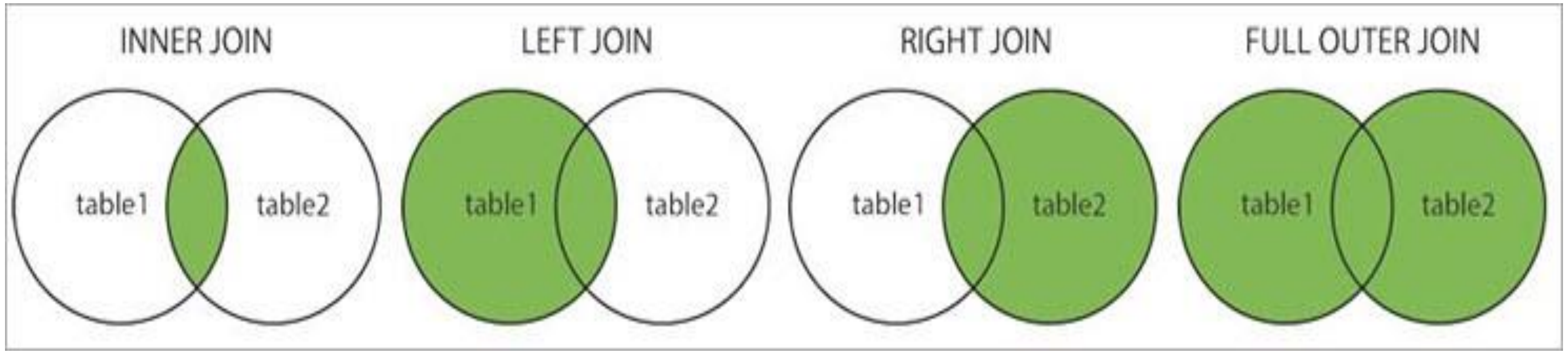
KING GRADUATE SCHOOL MONROE COLLEGE

Normalized JustLee Books Database…



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…



Subqueries and Their Uses…

- Subquery : a Query Inside Another Query…
- Used When a Query is Based on an Unknown Value…
- Requires SELECT and FROM Clauses…
- Must Be Enclosed in Parentheses…
- Place on Right Side of Comparison Operator…

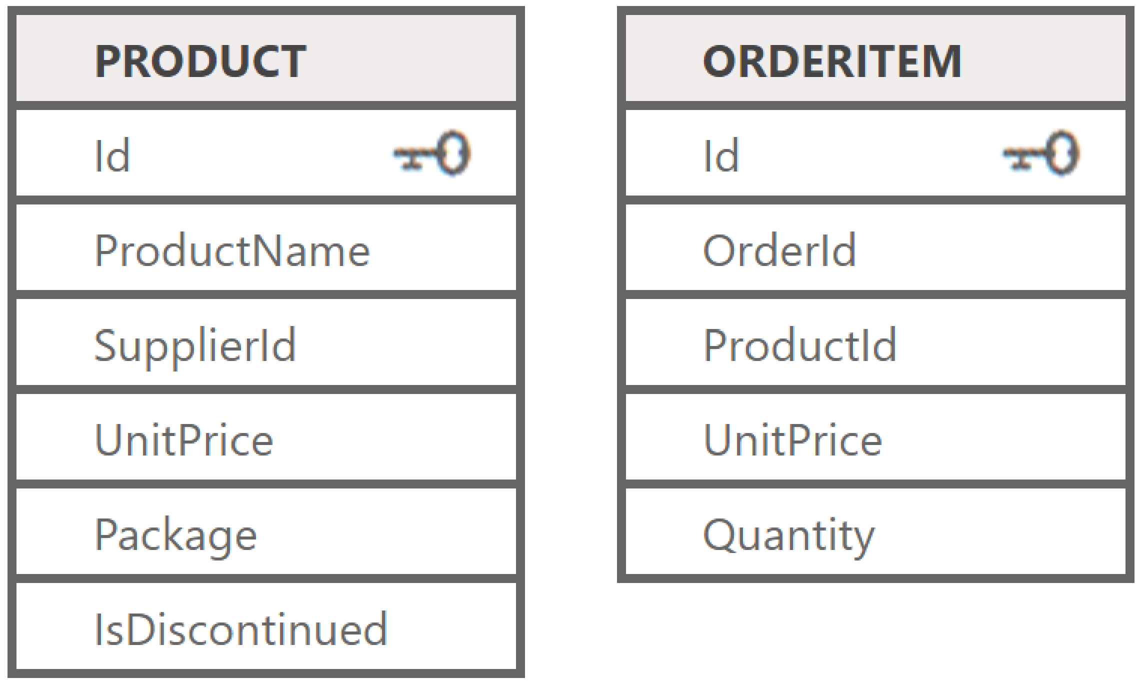
What?!...

- There is No General Syntax…Subqueries Are Regular Queries Placed Inside a Parenthesis…
- For Example…

```
SELECT column-names
FROM table-name1
WHERE value IN (SELECT column-name
                FROM table-name2
                WHERE condition);
```

Now I Get It…No Wait…What?!...

- Let’s Say We Wanted a List of Products With Order Quantities Greater Than 100…

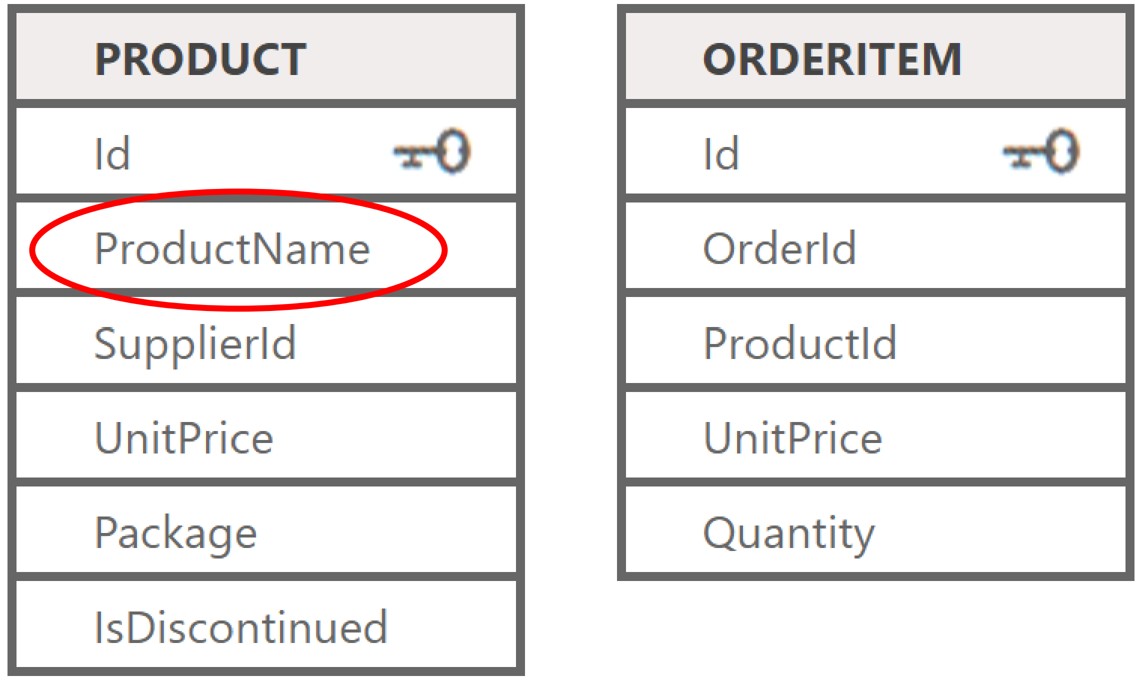


PRODUCT	
Id	🔑
ProductName	
SupplierId	
UnitPrice	
Package	
IsDiscontinued	

ORDERITEM	
Id	🔑
OrderId	
ProductId	
UnitPrice	
Quantity	

Now I Get It…No Wait…What?!...

- Let’s Say We Wanted a List of Products With Order Quantities Greater Than 100…
- Looking at the Tables, We Know That We Need to Select the ProductName From the PRODUCT Table…

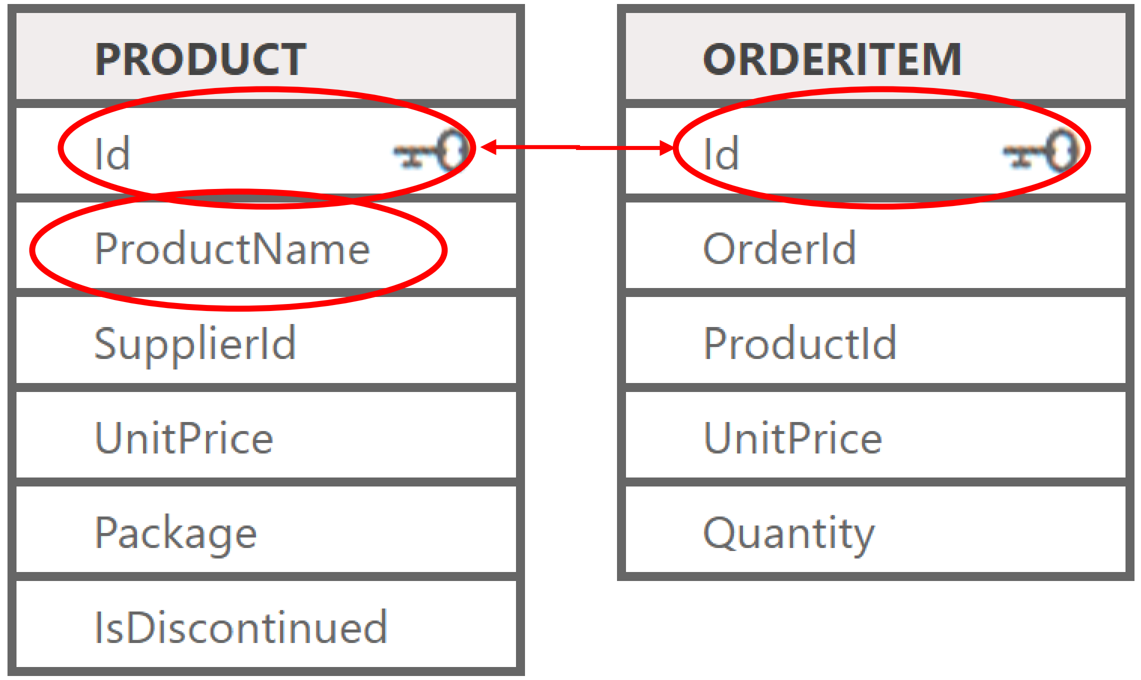


PRODUCT
Id 🔑
ProductName
SupplierId
UnitPrice
Package
IsDiscontinued

ORDERITEM
Id 🔑
OrderId
ProductId
UnitPrice
Quantity

Now I Get It…No Wait…What?!...

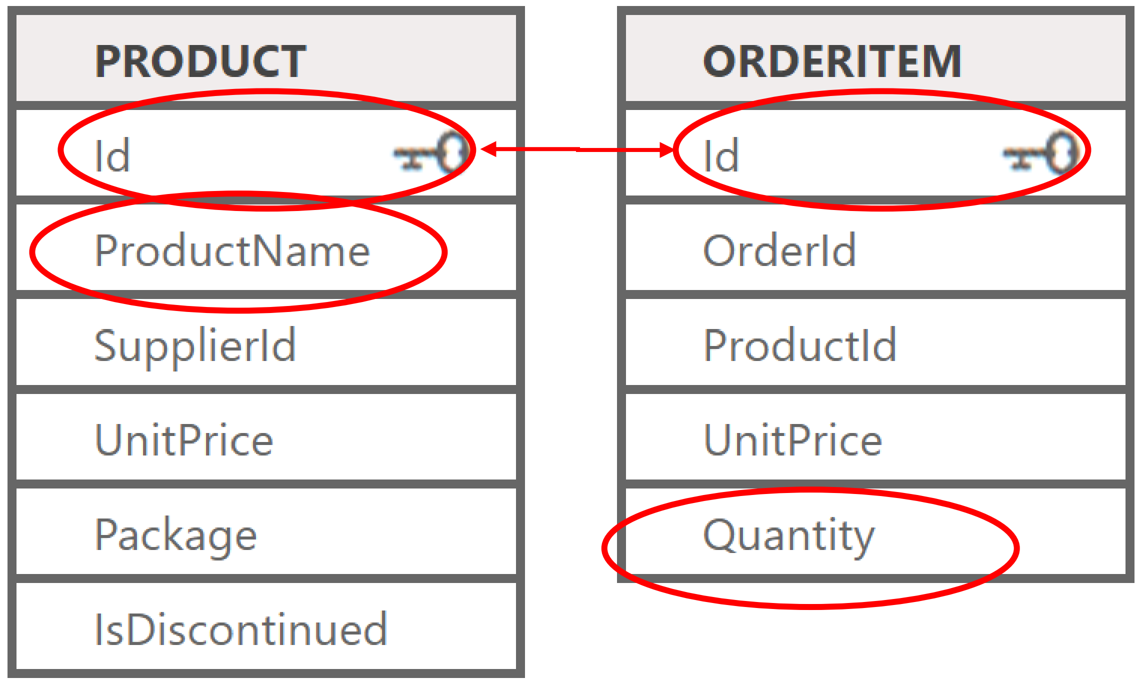
- Let's Say We Wanted a List of Products With Order Quantities Greater Than 100…
- Looking at the Tables, We Know That We Need to Select the ProductName From the PRODUCT Table…
- What We Don't Know Is the ID…



KING GRADUATE SCHOOL MONROE COLLEGE

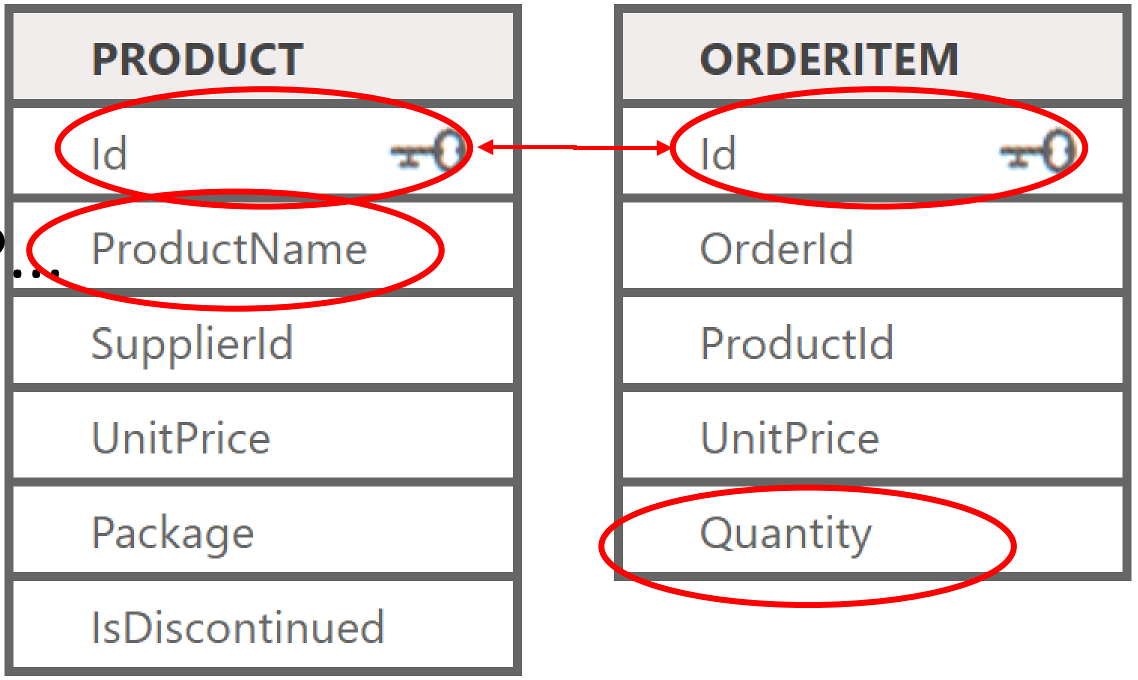
Now I Get It…No Wait…What?!...

- Let’s Say We Wanted a List of Products With Order Quantities Greater Than 100…
- Looking at the Tables, We Know That We Need to Select the ProductName From the PRODUCT Table…
- What We Don’t Know Is the ID…
- And We Don’t Know the Quantities…



Now I Get It…No Wait…What?!…

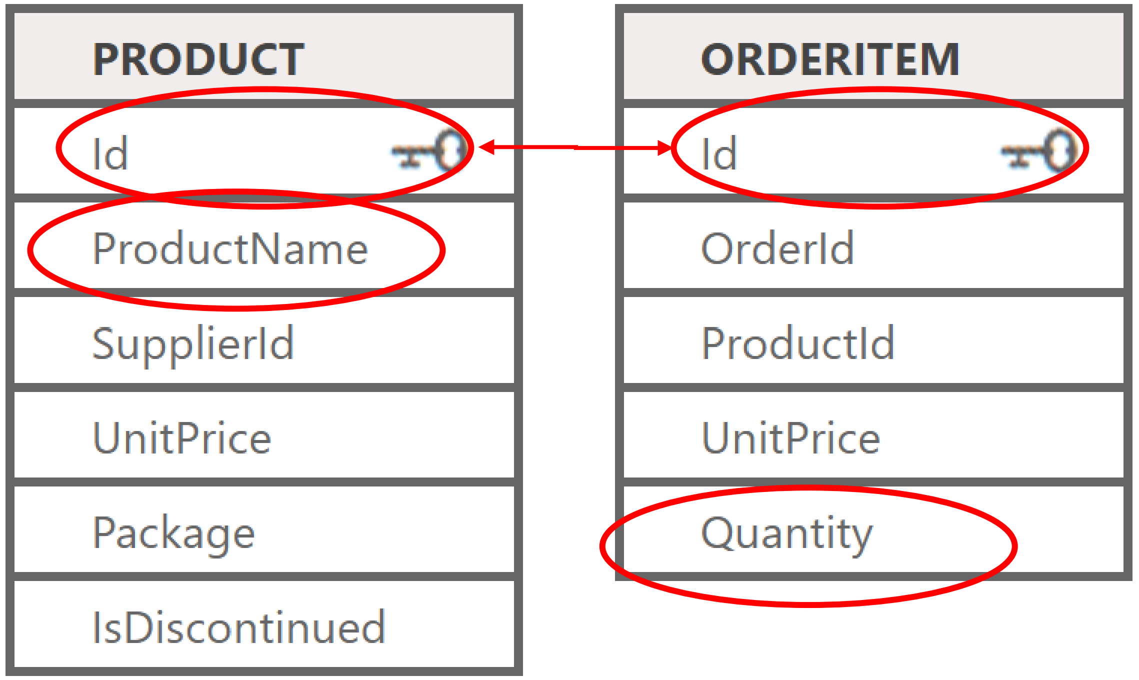
- Let's Say We Wanted a List of Products With Order Quantities Greater Than 100…
- Looking at the Tables, We Know That We Need to Select the ProductName From the PRODUCT Table…
- What We Don't Know Is the ID…
- And We Don't Know the Quantities…
- Could You TELL ME How to Get Them?…



KING GRADUATE SCHOOL MONROE COLLEGE

Now I Get It…No Wait…What?!...

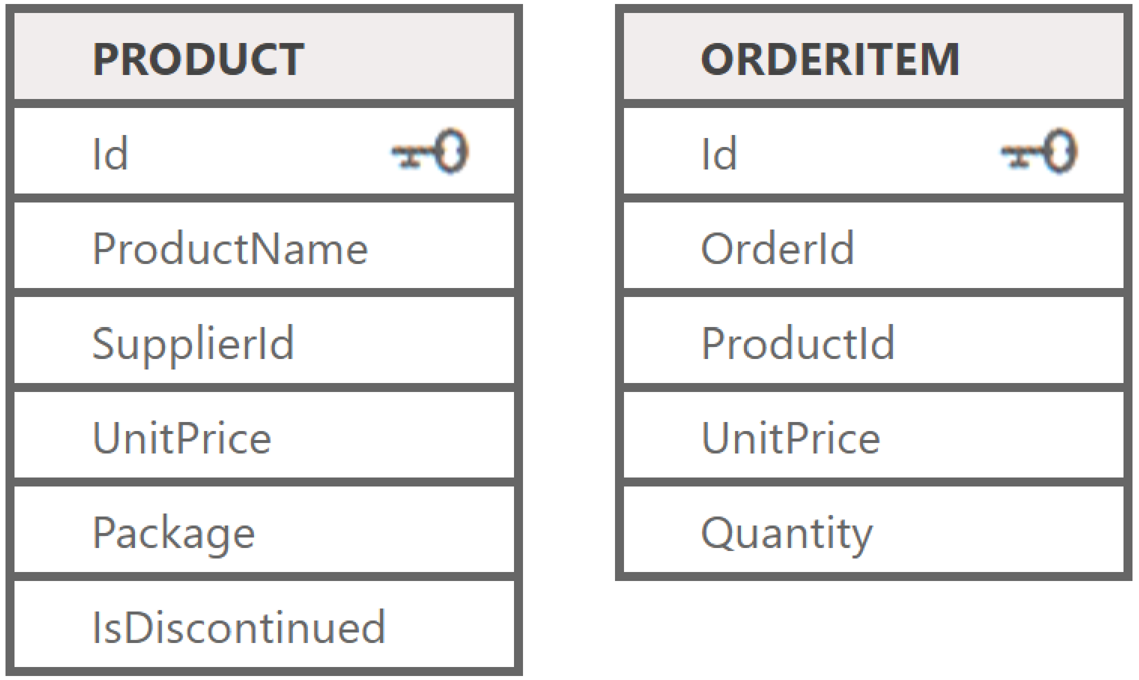
- Let's Say We Wanted a List of Products With Order Quantities Greater Than 100…
- Looking at the Tables, We Know That We Need to Select the ProductName From the PRODUCT Table…
- What We Don't Know Is the ID…
- And We Don't Know the Quantities...
- Could You TELL ME How to Get Them?...
- Bet You Could…



KING GRADUATE SCHOOL MONROE COLLEGE

Here's How…

- You Would Say to Go to the ORDERITEM Table and Select the ProductID for Products Where the Quantity is Greater Than 100…
- And I Bet You Can Easily Write That Query…
- That Query Would Look Like This…



PRODUCT
Id
ProductName
SupplierId
UnitPrice
Package
IsDiscontinued

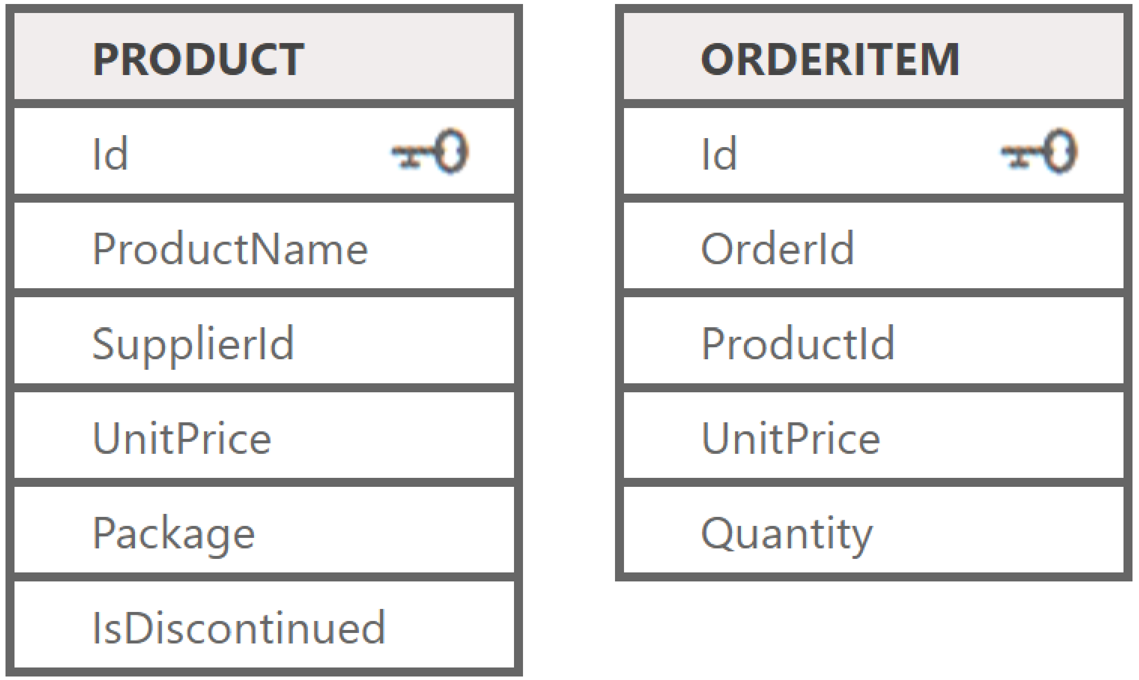
ORDERITEM
Id
OrderId
ProductId
UnitPrice
Quantity

KING GRADUATE SCHOOL MONROE COLLEGE

Here's How…

- You Would Say to Go to the ORDERITEM Table and Select the ProductID for Products Where the Quantity is Greater Than 100…
- And I Bet You Can Easily Write That Query…
- That Query Would Look Like This…

```
SELECT ProductID
FROM OrderItem
WHERE Quantity > 100;
```



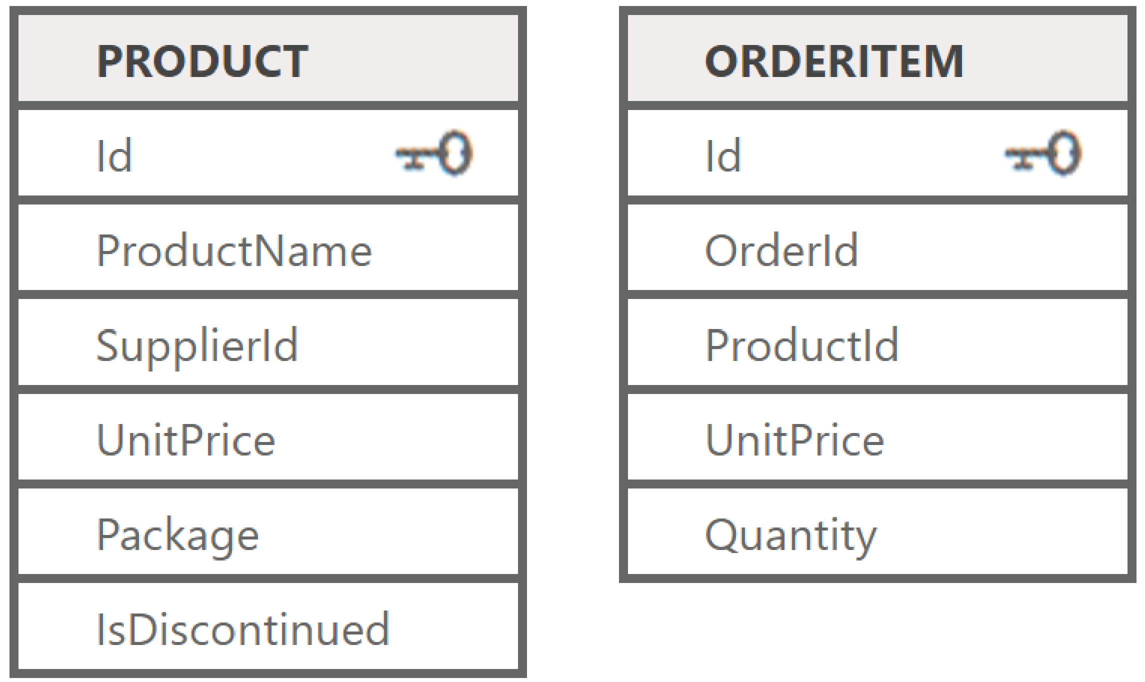
PRODUCT
Id
ProductName
SupplierId
UnitPrice
Package
IsDiscontinued

ORDERITEM
Id
OrderId
ProductId
UnitPrice
Quantity

KING GRADUATE SCHOOL MONROE COLLEGE

Ok…So Now What?…

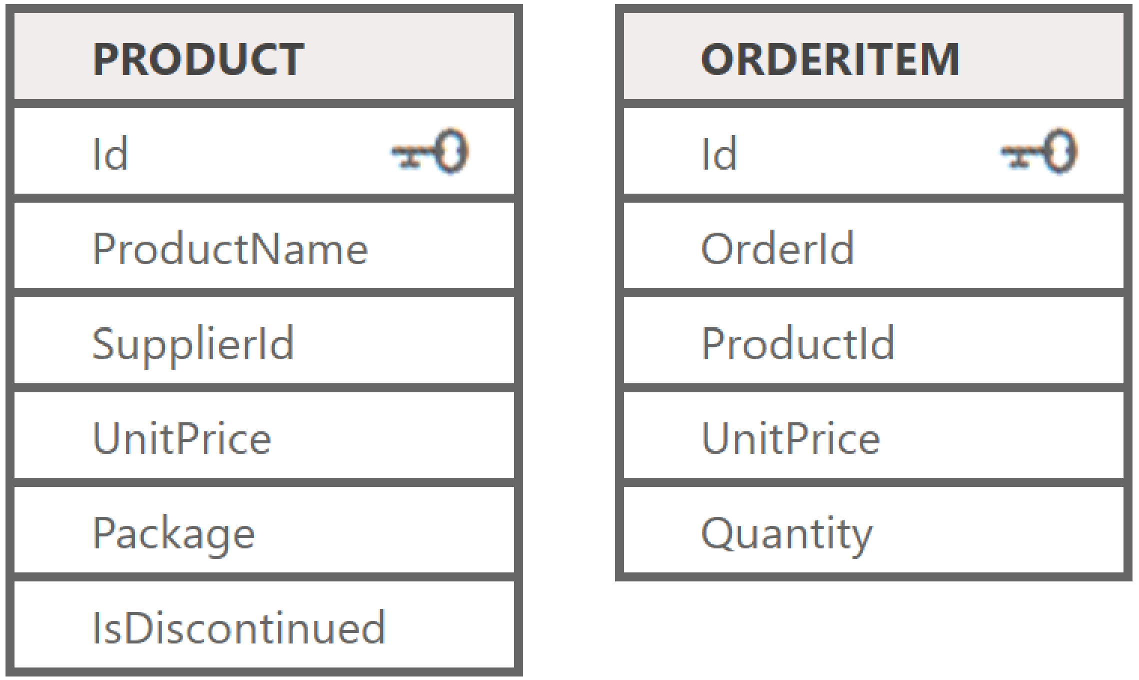
- Now We Need to Combine Everything That We Have Done So Far Into One Query…
- Your Query Will Look Like This…



Ok…So Now What?…

- Now We Need to Combine Everything That We Have Done So Far Into One Query…
- Your Query Will Look Like This…

```
SELECT ProductName
FROM   Product
WHERE Id IN (SELECT ProductID
                     FROM OrderItem
                     WHERE Quantity > 100);
```



PRODUCT
Id 🔑
ProductName
SupplierId
UnitPrice
Package
IsDiscontinued

ORDERITEM
Id 🔑
OrderId
ProductId
UnitPrice
Quantity

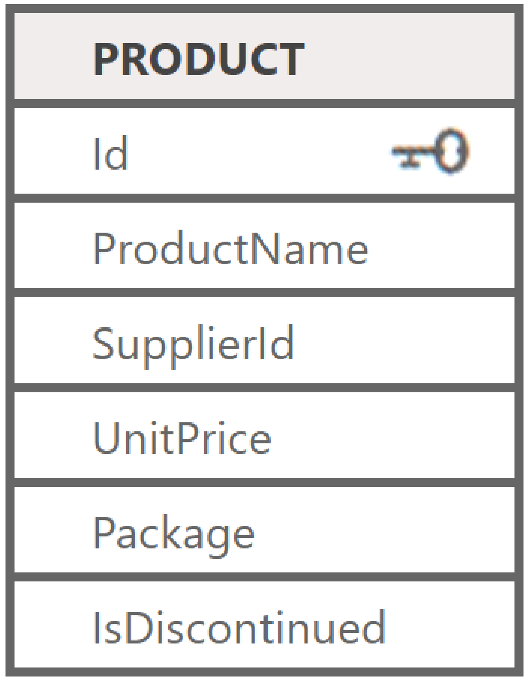
KING GRADUATE SCHOOL MONROE COLLEGE

Wait…Once More…

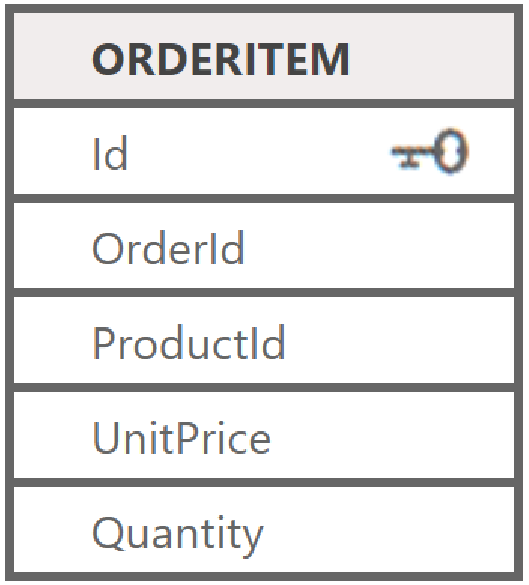
- We Know We Want to Show the Product Name for Products That Sold Over 100…
- But We Don’t Know What Those Products Are…And the Data That Tells Me That is in Another Table…
- So We Need to Build a Query With a Subquery…
- The First Part is What You Are Used to Seeing…SELECT this FROM there…
- But Now We Need to Figure Out the Where Clause…
- The Subquery Finds the IDs of the Items We Want…
- This Subquery is Put in the WHERE Clause…

That Leaves Us…

```
SELECT ProductName
FROM   Product
WHERE Id IN (SELECT ProductID
             FROM OrderItem
             WHERE Quantity > 100);
```



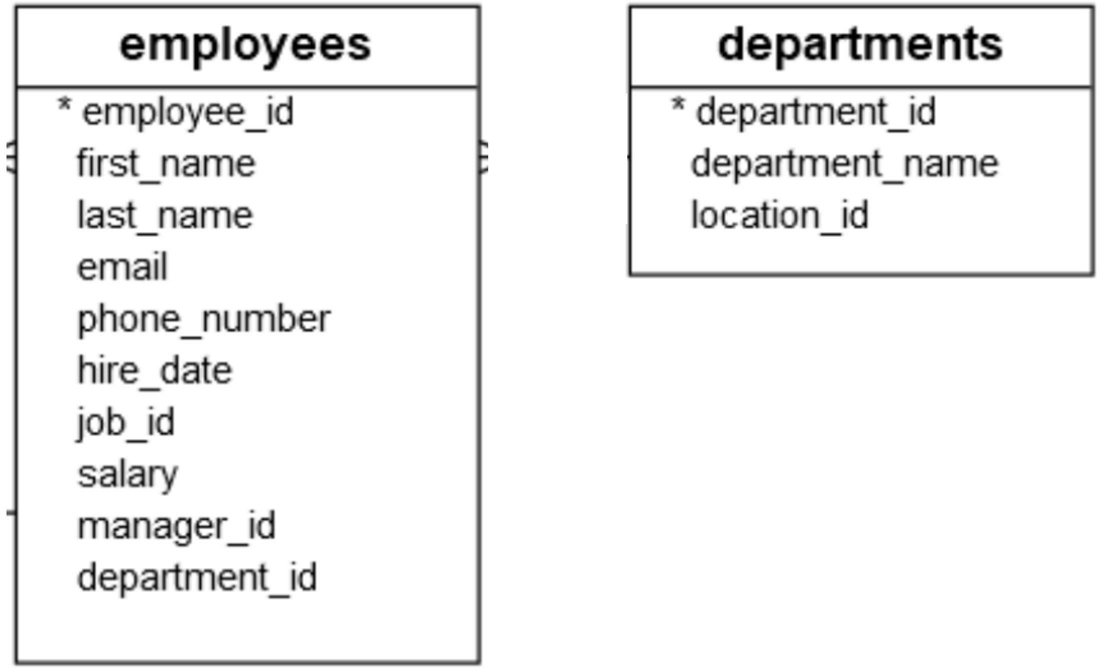
PRODUCT
Id 🔑
ProductName
SupplierId
UnitPrice
Package
IsDiscontinued



ORDERITEM
Id 🔑
OrderId
ProductId
UnitPrice
Quantity

How About This…

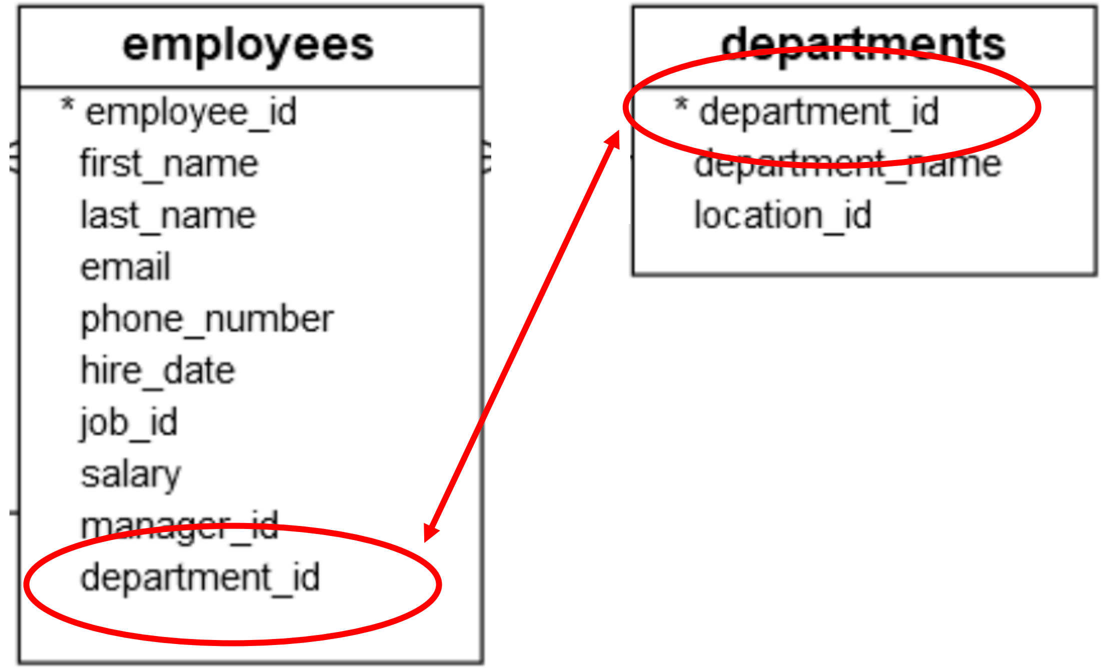
- Suppose You Have to Find All Employees Who Work in the Departments Located in LocationID 1700…



KING GRADUATE SCHOOL MONROE COLLEGE

How About This…

- Suppose You Have to Find All Employees Who Work in the Departments Located in LocationID 1700…
- Looking at These Two Tables, You Would Need to Match the DepartmentID in the EMPLOYEES table to the DepartmentID in the DEPARTMENTS Table…But…

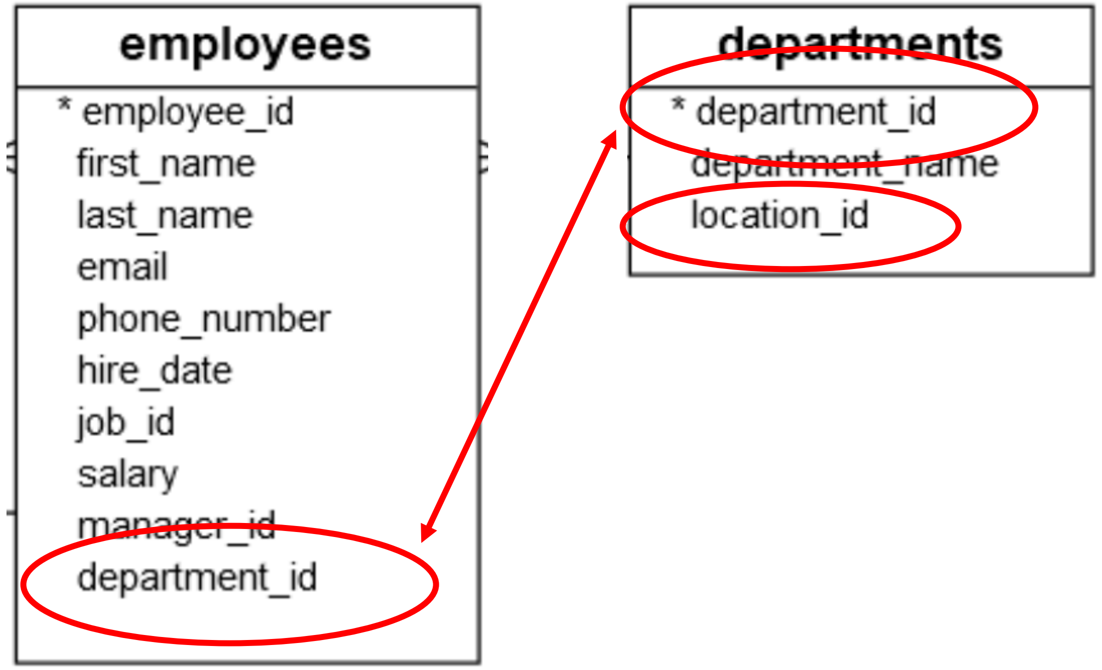


KING GRADUATE SCHOOL MONROE COLLEGE

How About This…

- Suppose You Have to Find All Employees Who Work in the Departments Located in LocationID 1700…
- Looking at These Two Tables, You Would Need to Match the DepartmentID in the EMPLOYEES table to the DepartmentID in the DEPARTMENTS Table…But…
- But Only for Departments With a

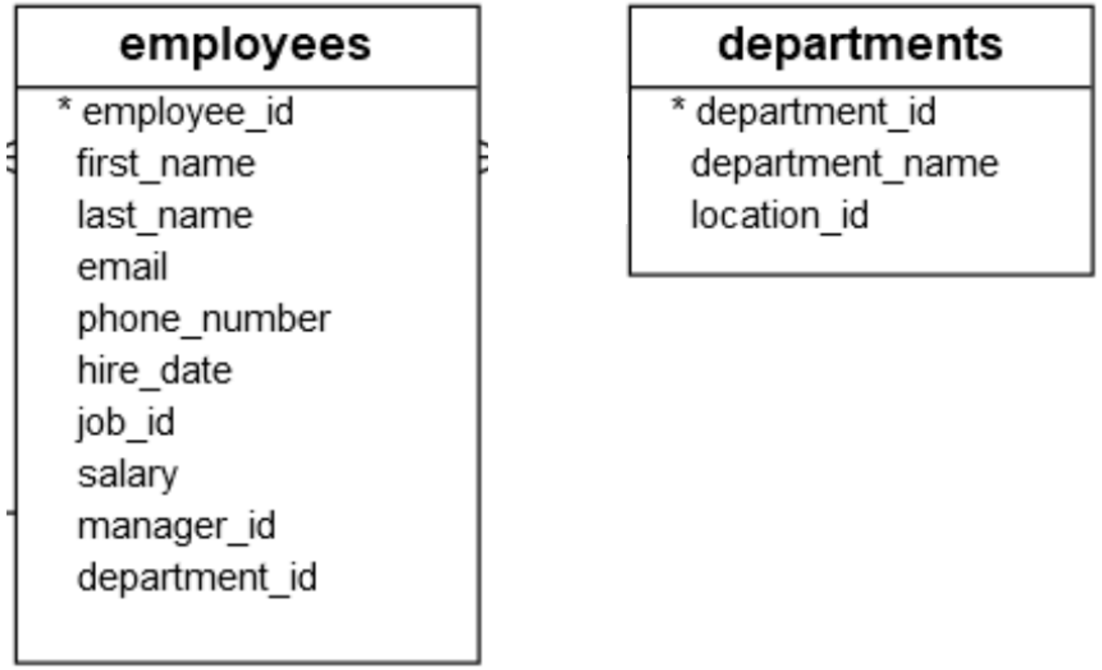
  LocationID of 1700…
- So Let's Tackle This…



KING GRADUATE SCHOOL MONROE COLLEGE

One Piece At a Time…

- Let's Start With Finding the Department That Are in Location 1700…
- Again, You Could Easily Write This Query…
- The Query Would Be…

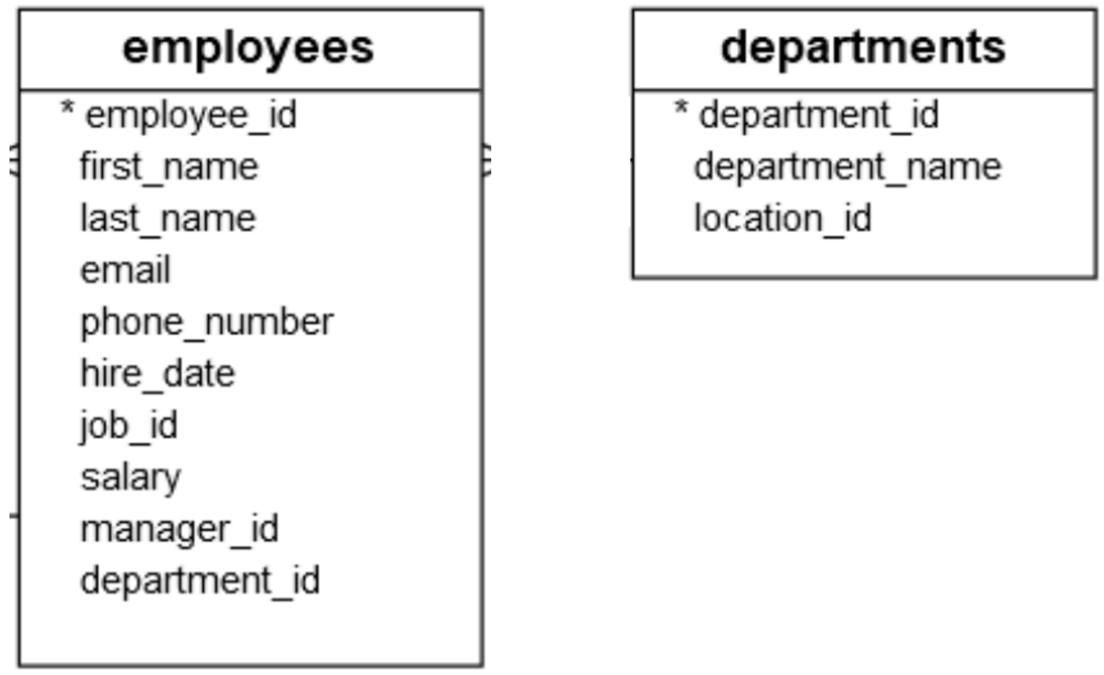


KING GRADUATE SCHOOL MONROE COLLEGE

One Piece At a Time…

- Let’s Start With Finding the Department That Are in Location 1700…
- Again, You Could Easily Write This Query…
- The Query Would Be…

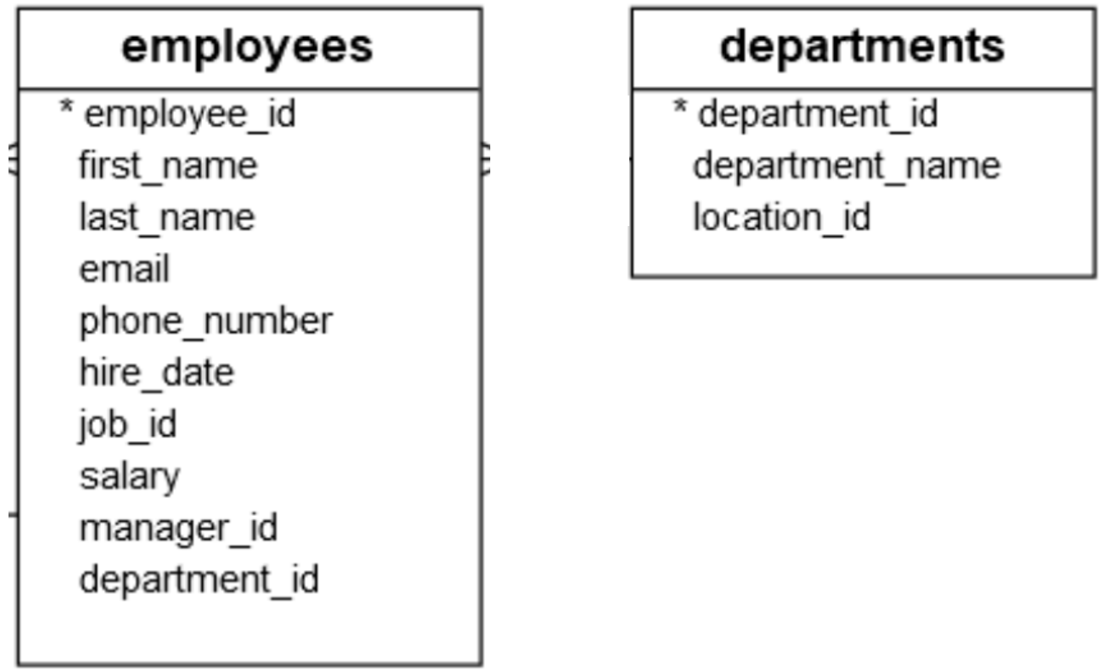
```
SELECT DepartmentID
FROM   Departments
WHERE LocationID = 1700;
```



KING GRADUATE SCHOOL MONROE COLLEGE

Now What About the Other Half…

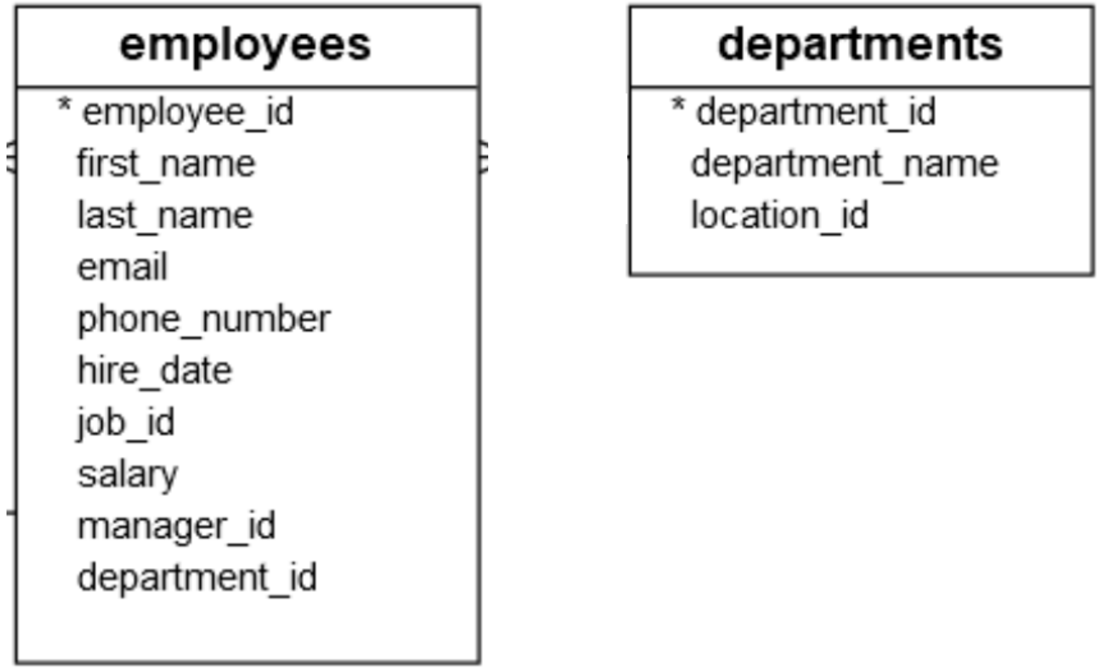
- We Know We Want the EmployID and Name From the EMPLOYEES Table…
- And We Know We Are Matching on DepartmentID…
- Again, You Could Easily Write This Query…
- The Query Would Be…



Now What About the Other Half…

- We Know We Want the EmployID and Name From the EMPLOYEES Table…
- And We Know We Are Matching on DepartmentID…
- Again, You Could Easily Write This Query…
- The Query Would Be…

```
SELECT EmployeeID, FirstName, LastName
FROM  Employees
WHERE DepartmentID IN …
```



I Think I Have It!…

- So Let’s Put the Two Pieces Together…

I Think I Have It!…

- So Let’s Put the Two Pieces Together…

```
SELECT EmployeeID, FirstName, LastName
FROM  Employees
WHERE DepartmentID IN
```

I Think I Have It!…

- So Let’s Put the Two Pieces Together…

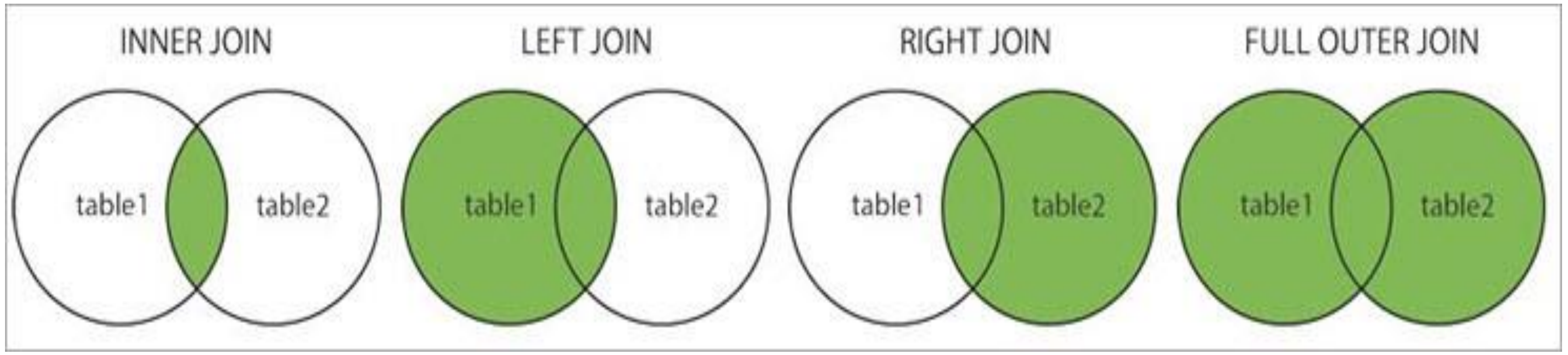
```
SELECT EmployeeID, FirstName, LastName
FROM  Employees
WHERE DepartmentID IN (SELECT DepartmentID
                       FROM   Departments
                       WHERE LocationID = 1700);
```

Remember…

- ANY QUERY YOU CAN WRITE CAN BE A SUBQUERY…
- By Looking at Each Piece Separately, You Should Be Able to Create These Relatively Easily…
- Just Remember, the Subquery Must :
    - ✓ Be Enclosed In Parenthesis…
    - ✓ Must Be on the Right Side of the Comparison Operator…

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

Types of Subqueries…

- **Single-Row :** Returns to the Outer Query One Row of Results That Consists of One Column…
- **Multiple-Row :** Returns to the Outer Query More Than One Row of Results…
- **Multiple-Column :** Returns to the Outer Query More Than One Column of Results…
- **Correlated :** References a Column in the Outer Query, and Executes the Subquery Once for Every Row in the Outer Query…
- **Uncorrelated :** Executes the Subquery First and Passes the Values to the Outer Query…

KING GRADUATE SCHOOL MONROE COLLEGE

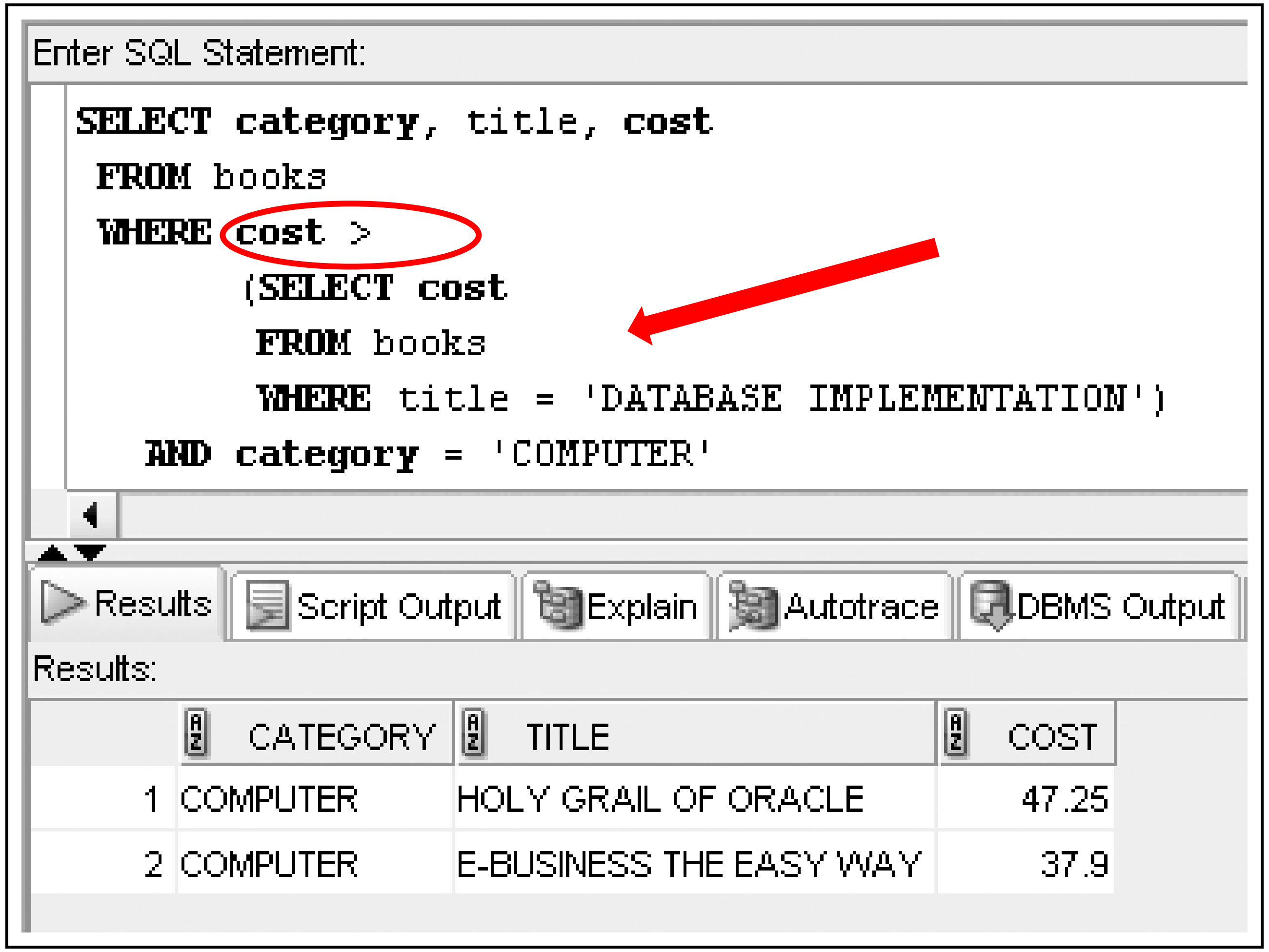
Single-Row Subqueries…

- Can Only Return One Result to the Outer Query…
- Can Be Used in a WHERE Clause, a HAVING Clause, or in the SELECT Clause…
- Operators Include =, >, <, >=, <=, < >…

KING GRADUATE SCHOOL MONROE COLLEGE

Single-Row Subquery in a WHERE Clause…

- Used for Comparison Against INDIVIDUAL Data…



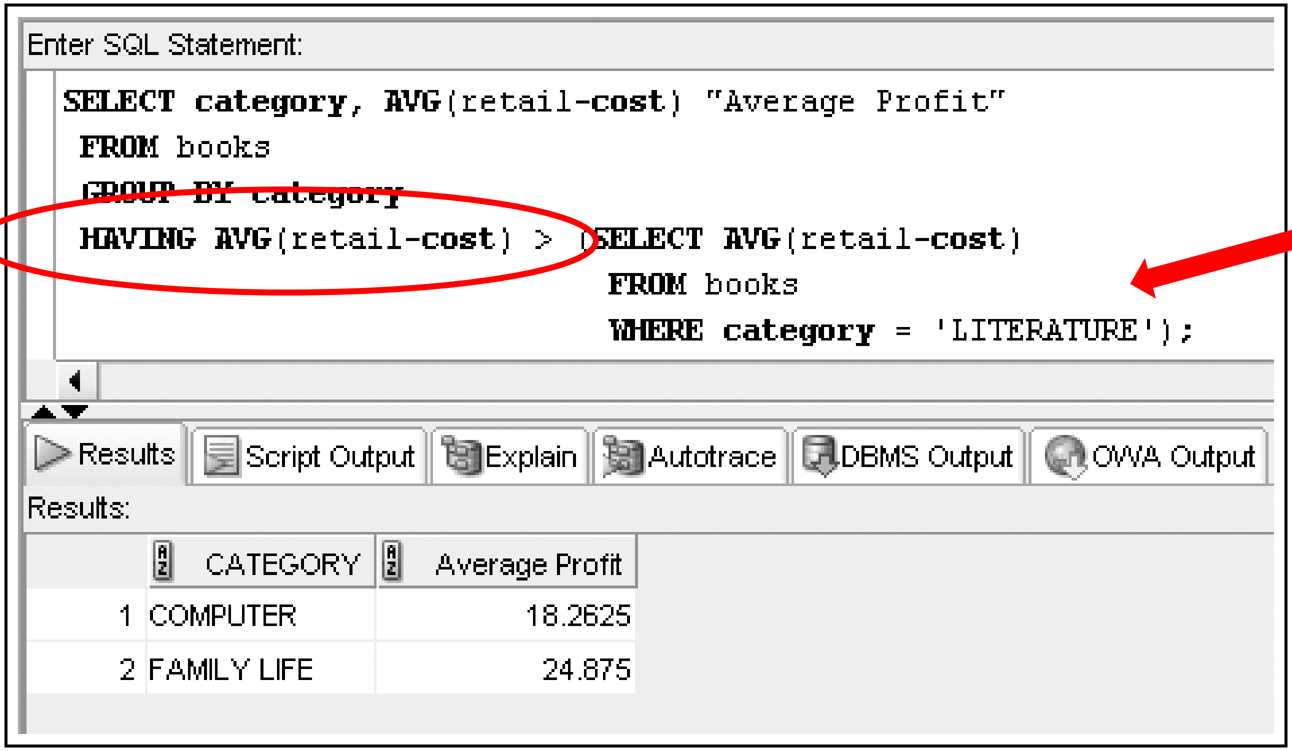
```
SELECT category, title, cost
 FROM books
 WHERE cost >
        (SELECT cost
         FROM books
         WHERE title = 'DATABASE IMPLEMENTATION')
   AND category = 'COMPUTER'
```

	CATEGORY	TITLE	COST
1	COMPUTER	HOLY GRAIL OF ORACLE	47.25
2	COMPUTER	E-BUSINESS THE EASY WAY	37.9

KING GRADUATE SCHOOL MONROE COLLEGE

Single-Row Subquery in a HAVING Clause…

- Required When Returned Value is Compared to Grouped Data…



Enter SQL Statement:

```
SELECT category, AVG(retail-cost) "Average Profit"
 FROM books
 GROUP BY category
 HAVING AVG(retail-cost) > (SELECT AVG(retail-cost)
                             FROM books
                             WHERE category = 'LITERATURE');
```

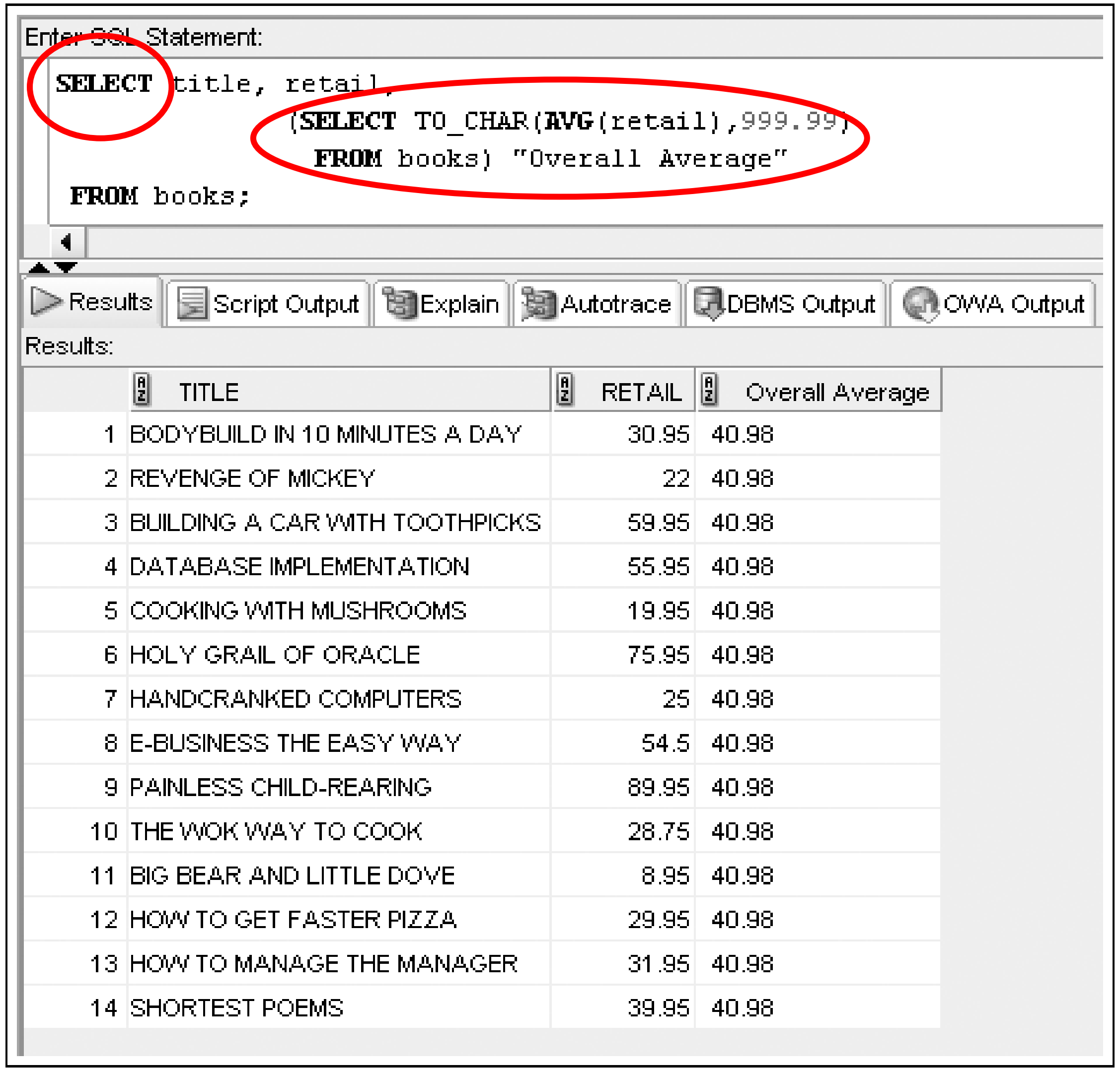
Results:

```
   CATEGORY      Average Profit
1  COMPUTER             18.2625
2  FAMILY LIFE           24.875
```

KING GRADUATE SCHOOL MONROE COLLEGE

Single-Row Subquery in a SELECT Clause…

- Replicates Subquery Value For Each Row Displayed…



```
SELECT title, retail,
            (SELECT TO_CHAR(AVG(retail),999.99)
             FROM books) "Overall Average"
  FROM books;
```

	TITLE	RETAIL	Overall Average
1	BODYBUILD IN 10 MINUTES A DAY	30.95	40.98
2	REVENGE OF MICKEY	22	40.98
3	BUILDING A CAR WITH TOOTHPICKS	59.95	40.98
4	DATABASE IMPLEMENTATION	55.95	40.98
5	COOKING WITH MUSHROOMS	19.95	40.98
6	HOLY GRAIL OF ORACLE	75.95	40.98
7	HANDCRANKED COMPUTERS	25	40.98
8	E-BUSINESS THE EASY WAY	54.5	40.98
9	PAINLESS CHILD-REARING	89.95	40.98
10	THE WOK WAY TO COOK	28.75	40.98
11	BIG BEAR AND LITTLE DOVE	8.95	40.98
12	HOW TO GET FASTER PIZZA	29.95	40.98
13	HOW TO MANAGE THE MANAGER	31.95	40.98
14	SHORTEST POEMS	39.95	40.98

KING GRADUATE SCHOOL MONROE COLLEGE

Multiple-Row Subqueries…

- Return More Than One Row of Results…
- Can Be Used in a WHERE Clause, or a HAVING Clause…
- Require Use of IN, ANY, ALL, or EXISTS Operators…

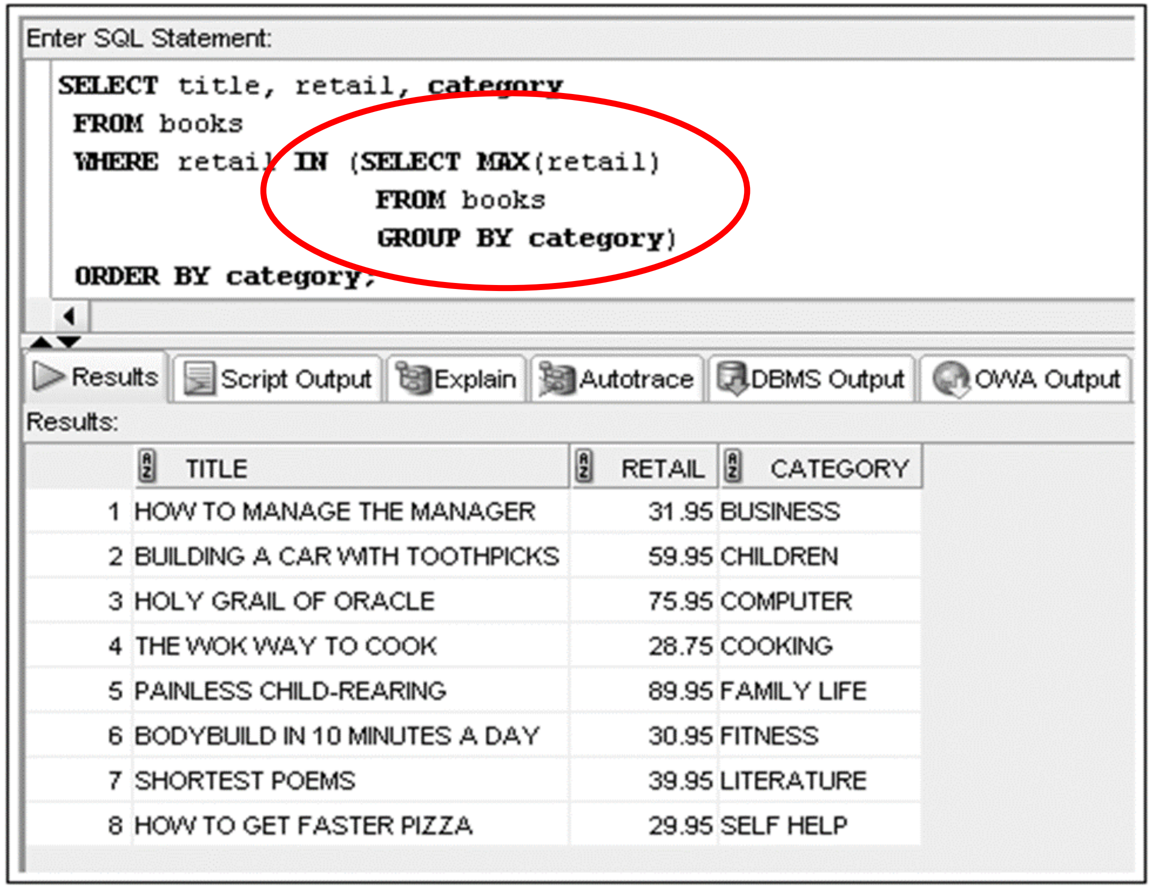
KING GRADUATE SCHOOL MONROE COLLEGE

ANY and ALL Operators…

- Combine With Arithmetic Operators…
- **>ALL :** More Than the Highest Value Returned by the Subquery…
- **<ALL :** Less Than the Lowest Value Returned by the Subquery…
- **<ANY :** Less Than the Highest Value Returned by the Subquery…
- **>ANY :** More Than the Lowest Value Returned by the Subquery…
- **=ANY :** Equal to Any Value Returned by the Subquery…Same as IN…

KING GRADUATE SCHOOL MONROE COLLEGE

Multiple-Row Subquery in a WHERE Clause…

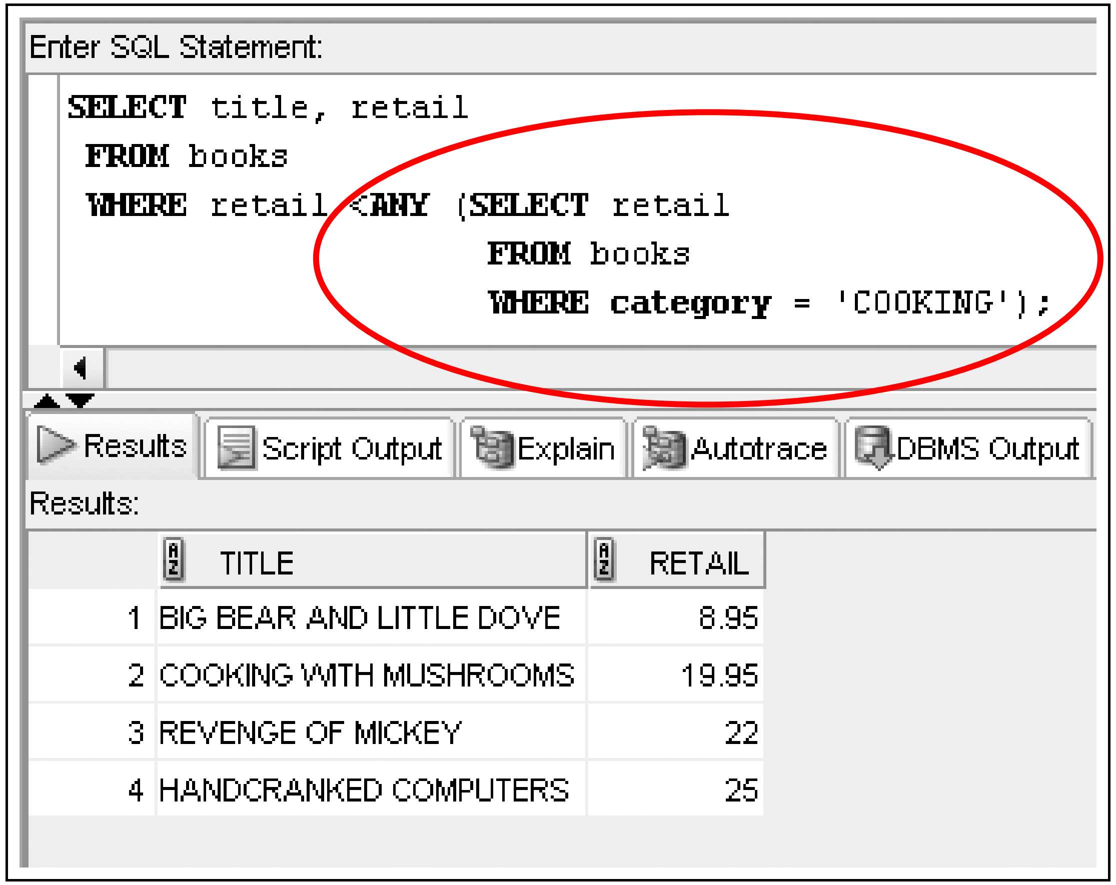


Enter SQL Statement:

```
SELECT title, retail, category
 FROM books
 WHERE retail IN (SELECT MAX(retail)
                    FROM books
                    GROUP BY category)
 ORDER BY category;
```

Results:

	TITLE	RETAIL	CATEGORY
1	HOW TO MANAGE THE MANAGER	31.95	BUSINESS
2	BUILDING A CAR WITH TOOTHPICKS	59.95	CHILDREN
3	HOLY GRAIL OF ORACLE	75.95	COMPUTER
4	THE WOK WAY TO COOK	28.75	COOKING
5	PAINLESS CHILD-REARING	89.95	FAMILY LIFE
6	BODYBUILD IN 10 MINUTES A DAY	30.95	FITNESS
7	SHORTEST POEMS	39.95	LITERATURE
8	HOW TO GET FASTER PIZZA	29.95	SELF HELP



Enter SQL Statement:

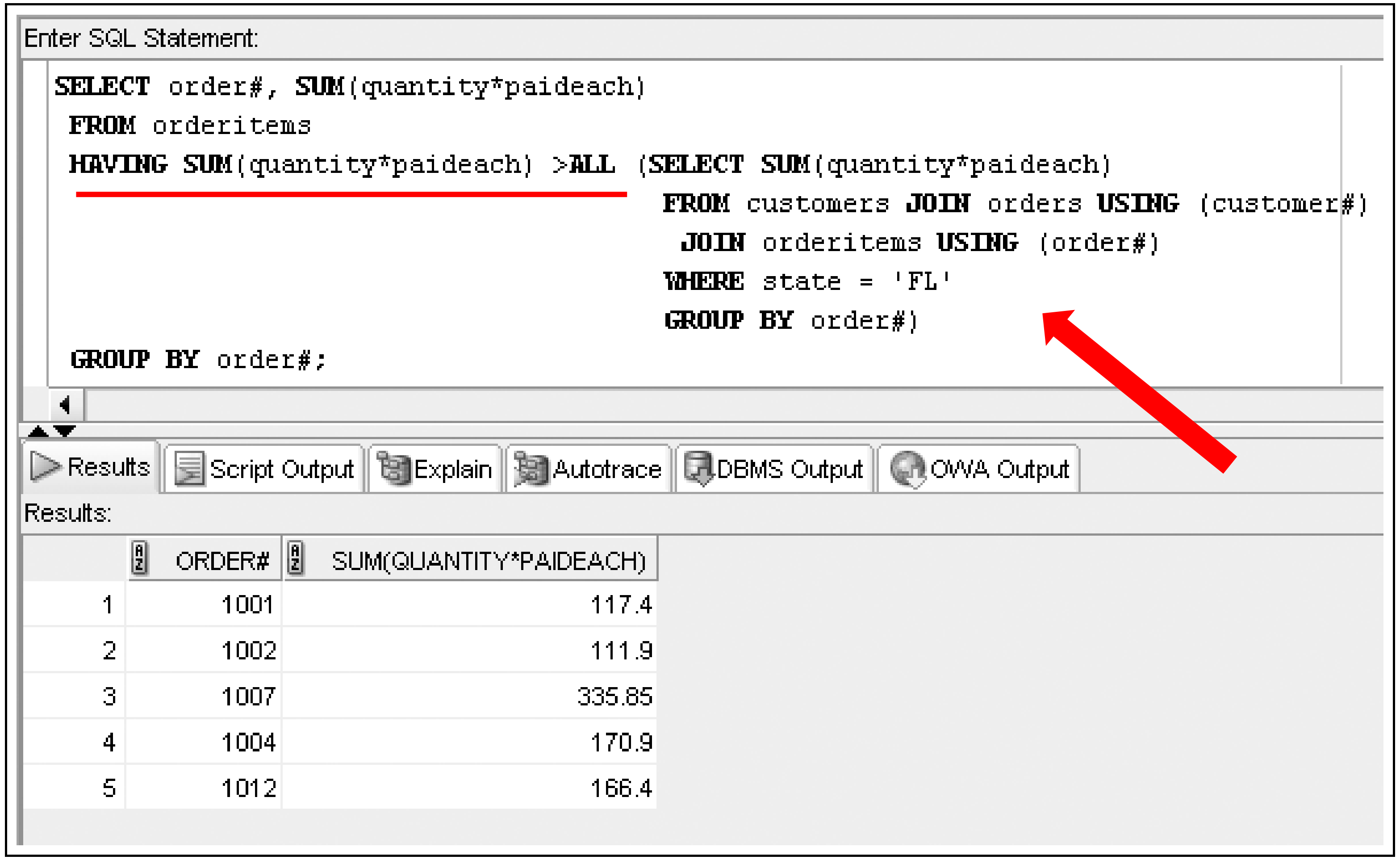
```
SELECT title, retail
 FROM books
 WHERE retail <ANY (SELECT retail
                     FROM books
                     WHERE category = 'COOKING');
```

Results:

	TITLE	RETAIL
1	BIG BEAR AND LITTLE DOVE	8.95
2	COOKING WITH MUSHROOMS	19.95
3	REVENGE OF MICKEY	22
4	HANDCRANKED COMPUTERS	25

KING GRADUATE SCHOOL MONROE COLLEGE

Multiple-Row Subquery in a HAVING Clause…


Enter SQL Statement:

```
SELECT order#, SUM(quantity*paideach)
 FROM orderitems
 HAVING SUM(quantity*paideach) >ALL (SELECT SUM(quantity*paideach)
                                       FROM customers JOIN orders USING (customer#)
                                        JOIN orderitems USING (order#)
                                       WHERE state = 'FL'
                                       GROUP BY order#)
 GROUP BY order#;
```

Results:

	ORDER#	SUM(QUANTITY*PAIDEACH)
1	1001	117.4
2	1002	111.9
3	1007	335.85
4	1004	170.9
5	1012	166.4

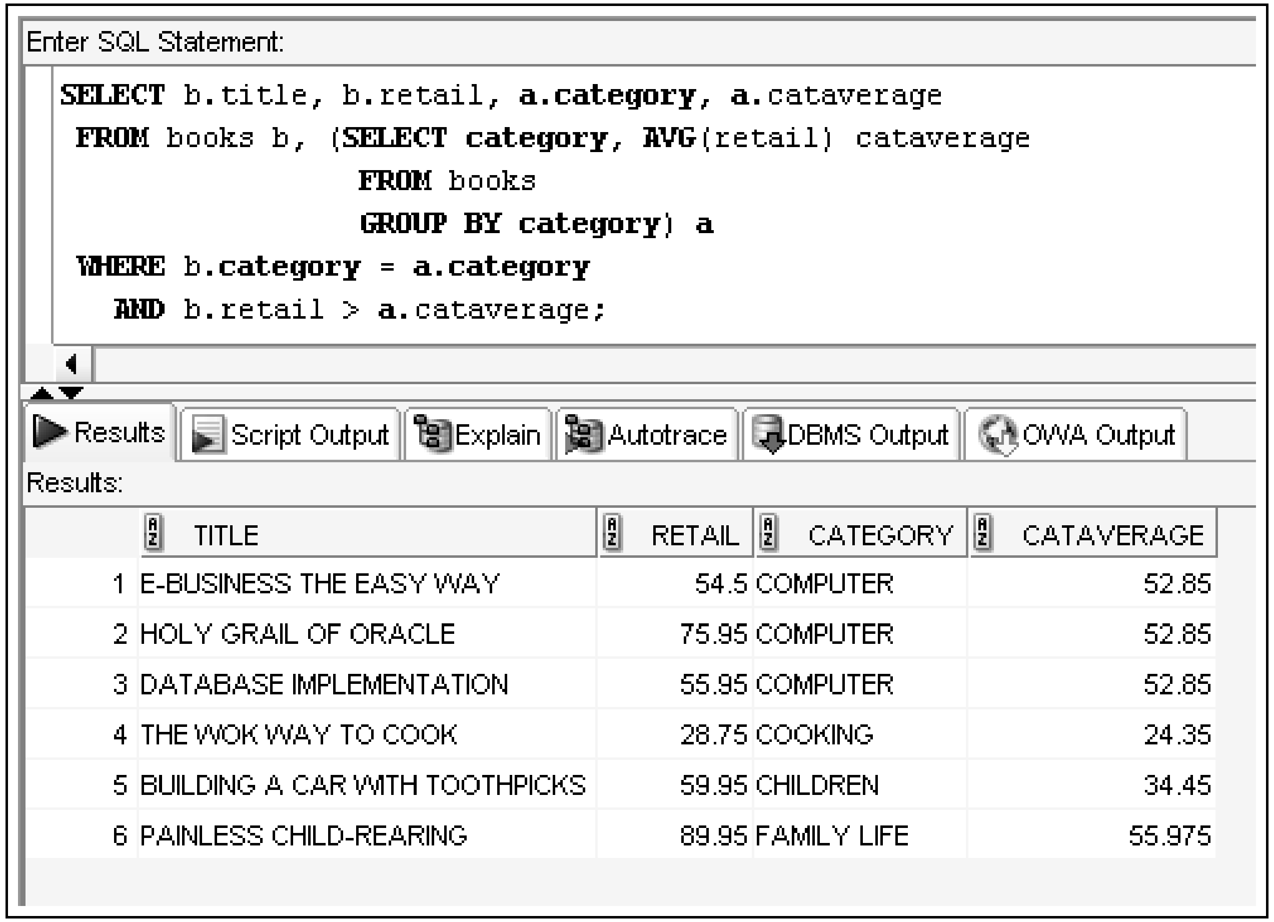
Multiple-Column Subqueries…

- Return More Than One Column in Results…
- Can Return More Than One Row…
- Column List On the Left Side of Operator Must Be In Parentheses…
- Use the IN Operator For WHERE and HAVING Clauses…

KING GRADUATE SCHOOL MONROE COLLEGE

Multiple-Column Subquery in a FROM Clause…

- Creates a Temporary Table…



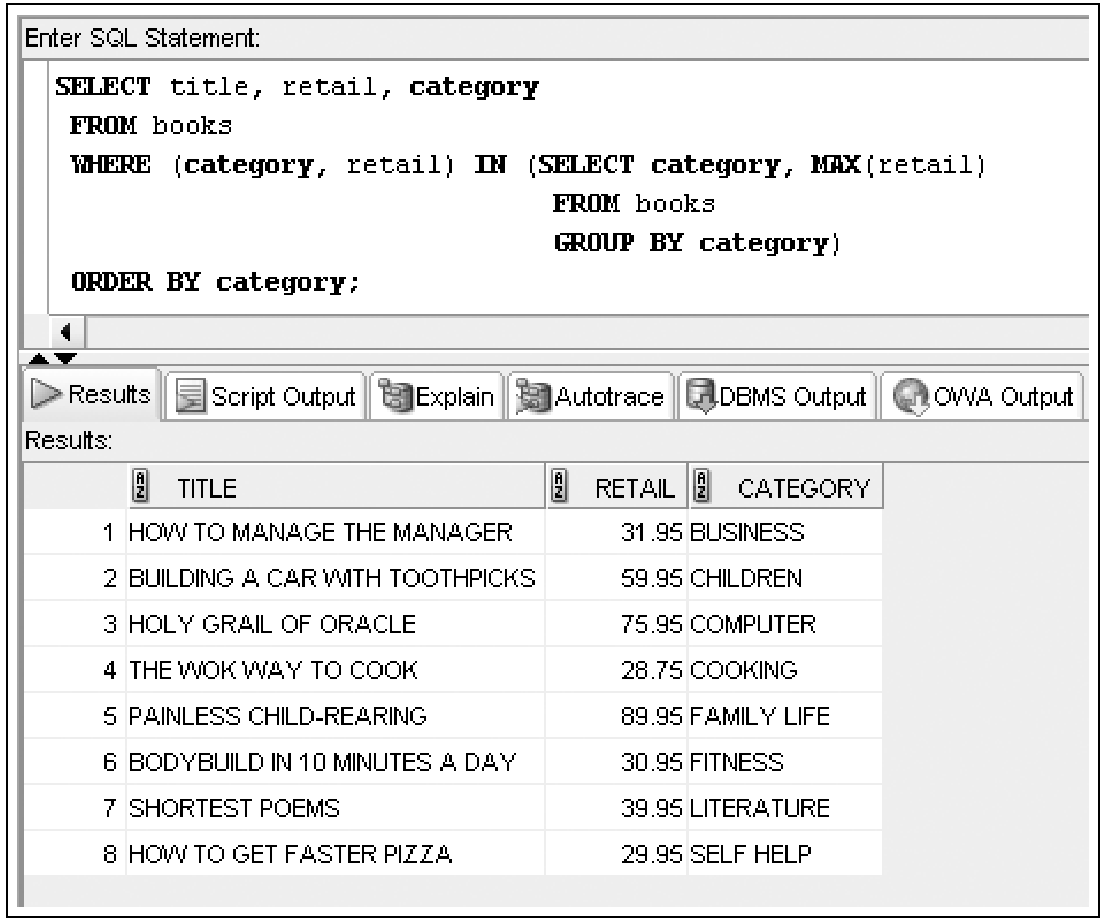
```
SELECT b.title, b.retail, a.category, a.cataverage
 FROM books b, (SELECT category, AVG(retail) cataverage
                  FROM books
                  GROUP BY category) a
 WHERE b.category = a.category
   AND b.retail > a.cataverage;
```

	TITLE	RETAIL	CATEGORY	CATAVERAGE
1	E-BUSINESS THE EASY WAY	54.5	COMPUTER	52.85
2	HOLY GRAIL OF ORACLE	75.95	COMPUTER	52.85
3	DATABASE IMPLEMENTATION	55.95	COMPUTER	52.85
4	THE WOK WAY TO COOK	28.75	COOKING	24.35
5	BUILDING A CAR WITH TOOTHPICKS	59.95	CHILDREN	34.45
6	PAINLESS CHILD-REARING	89.95	FAMILY LIFE	55.975

KING GRADUATE SCHOOL MONROE COLLEGE

Multiple-Column Subquery in a WHERE Clause…

- Returns Multiple Columns For Evaluation…



Enter SQL Statement:

```
SELECT title, retail, category
 FROM books
 WHERE (category, retail) IN (SELECT category, MAX(retail)
                                 FROM books
                                 GROUP BY category)
 ORDER BY category;
```

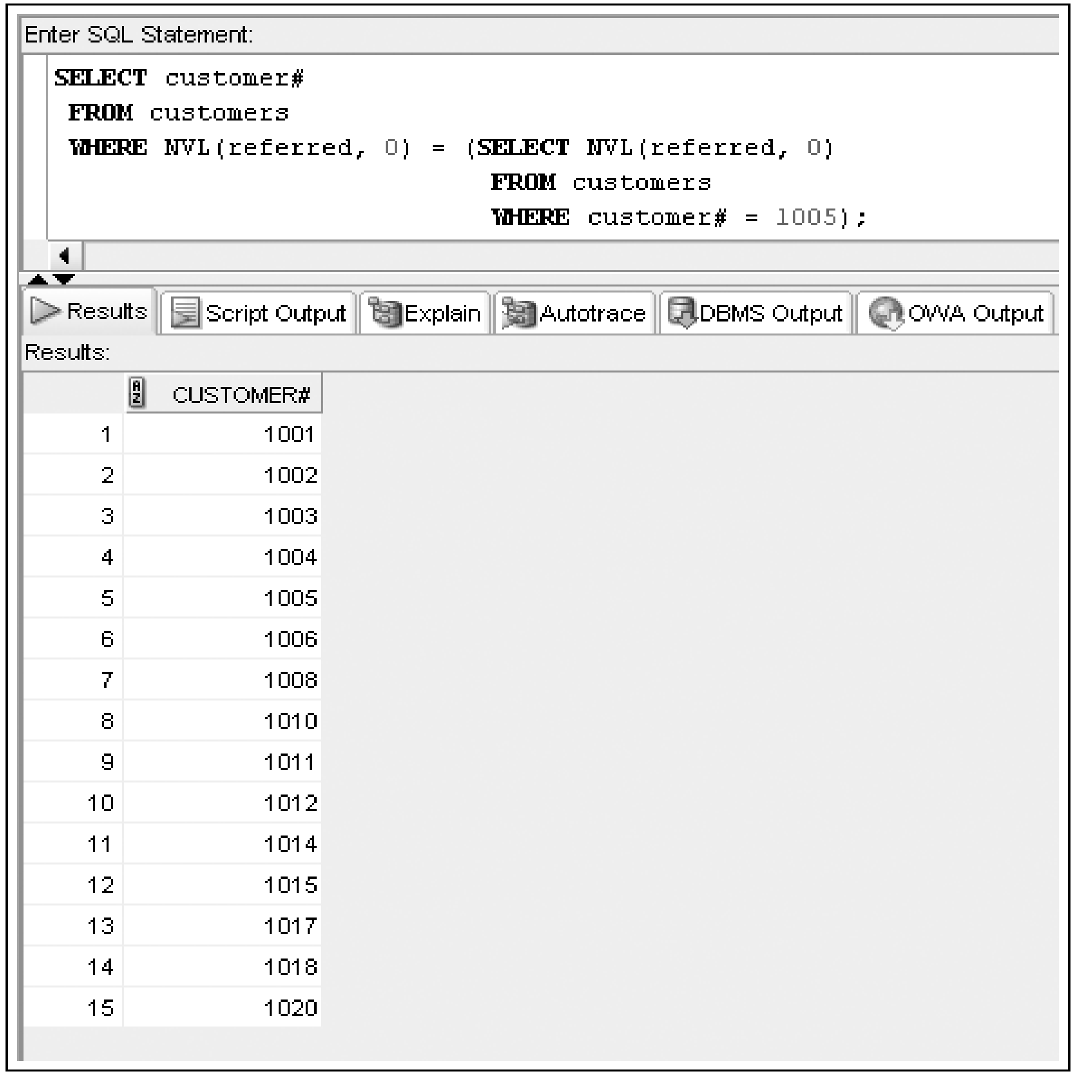
Results:

	TITLE	RETAIL	CATEGORY
1	HOW TO MANAGE THE MANAGER	31.95	BUSINESS
2	BUILDING A CAR WITH TOOTHPICKS	59.95	CHILDREN
3	HOLY GRAIL OF ORACLE	75.95	COMPUTER
4	THE WOK WAY TO COOK	28.75	COOKING
5	PAINLESS CHILD-REARING	89.95	FAMILY LIFE
6	BODYBUILD IN 10 MINUTES A DAY	30.95	FITNESS
7	SHORTEST POEMS	39.95	LITERATURE
8	HOW TO GET FASTER PIZZA	29.95	SELF HELP

KING GRADUATE SCHOOL MONROE COLLEGE

NULL Values…

- When a Subquery Might Return NULL Values, Use NVL Function…



```
SELECT customer#
 FROM customers
 WHERE NVL(referred, 0) = (SELECT NVL(referred, 0)
                            FROM customers
                            WHERE customer# = 1005);
```

	CUSTOMER#
1	1001
2	1002
3	1003
4	1004
5	1005
6	1006
7	1008
8	1010
9	1011
10	1012
11	1014
12	1015
13	1017
14	1018
15	1020

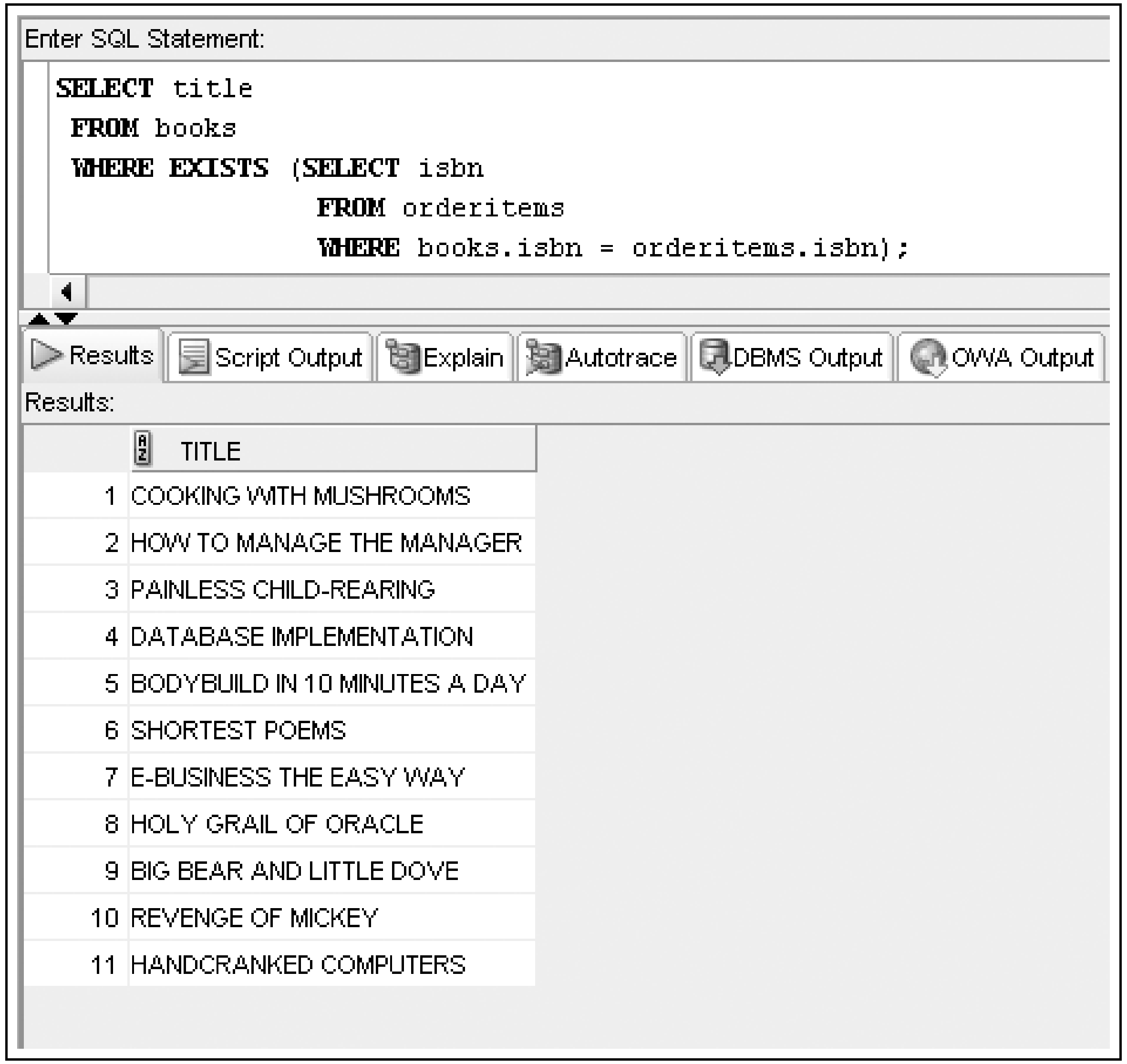
KING GRADUATE SCHOOL MONROE COLLEGE

Uncorrelated Subqueries…

- Processing Sequence :
  - ✓Inner Query Is Executed First…
  - ✓Result Is Passed to Outer Query…
  - ✓Outer Query is Executed…

Correlated Subqueries…

- Inner Query is Executed Once For Each Row Processed By the Outer Query…
- Inner Query References the Row Contained in the Outer Query…



```
SELECT title
 FROM books
 WHERE EXISTS (SELECT isbn
                 FROM orderitems
                 WHERE books.isbn = orderitems.isbn);
```

	TITLE
1	COOKING WITH MUSHROOMS
2	HOW TO MANAGE THE MANAGER
3	PAINLESS CHILD-REARING
4	DATABASE IMPLEMENTATION
5	BODYBUILD IN 10 MINUTES A DAY
6	SHORTEST POEMS
7	E-BUSINESS THE EASY WAY
8	HOLY GRAIL OF ORACLE
9	BIG BEAR AND LITTLE DOVE
10	REVENGE OF MICKEY
11	HANDCRANKED COMPUTERS

KING GRADUATE SCHOOL MONROE COLLEGE

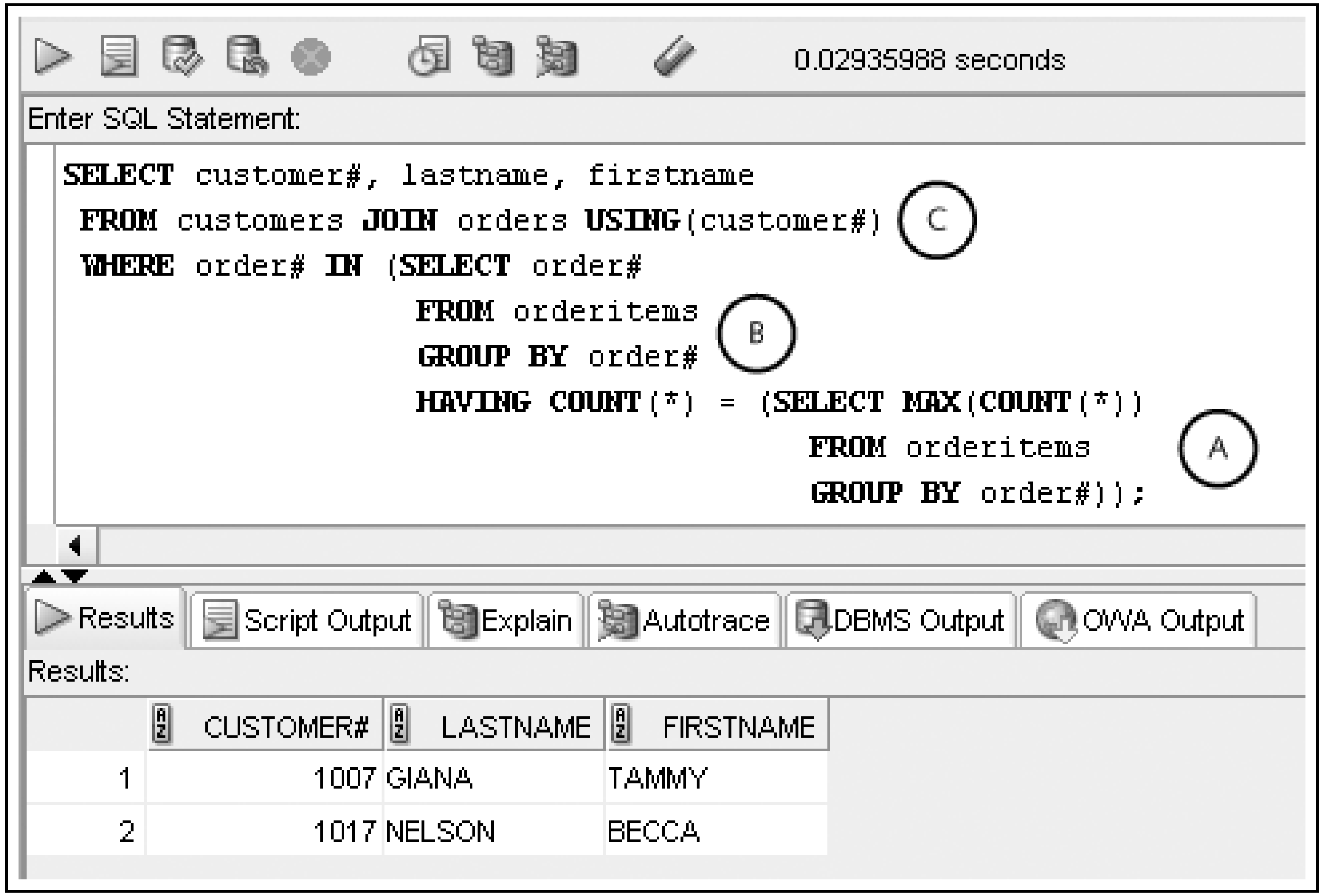
Nested Subqueries – Part 1…

- Maximum of 255 Subqueries If Nested in the WHERE Clause…
- No Limit If Nested in the FROM Clause…
- Innermost Subquery is Resolved First…Then the Next Level, and the Next, Etc….

KING GRADUATE SCHOOL MONROE COLLEGE

Nested Subqueries – Part 2…

- Innermost is Resolved First (A)…Then the Second Level (B)…Then the Outer Query (C)…



```
SELECT customer#, lastname, firstname
 FROM customers JOIN orders USING(customer#)
 WHERE order# IN (SELECT order#
                    FROM orderitems
                    GROUP BY order#
                    HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                                        FROM orderitems
                                        GROUP BY order#));
```

	CUSTOMER#	LASTNAME	FIRSTNAME
1	1007	GIANA	TAMMY
2	1017	NELSON	BECCA

KING GRADUATE SCHOOL MONROE COLLEGE

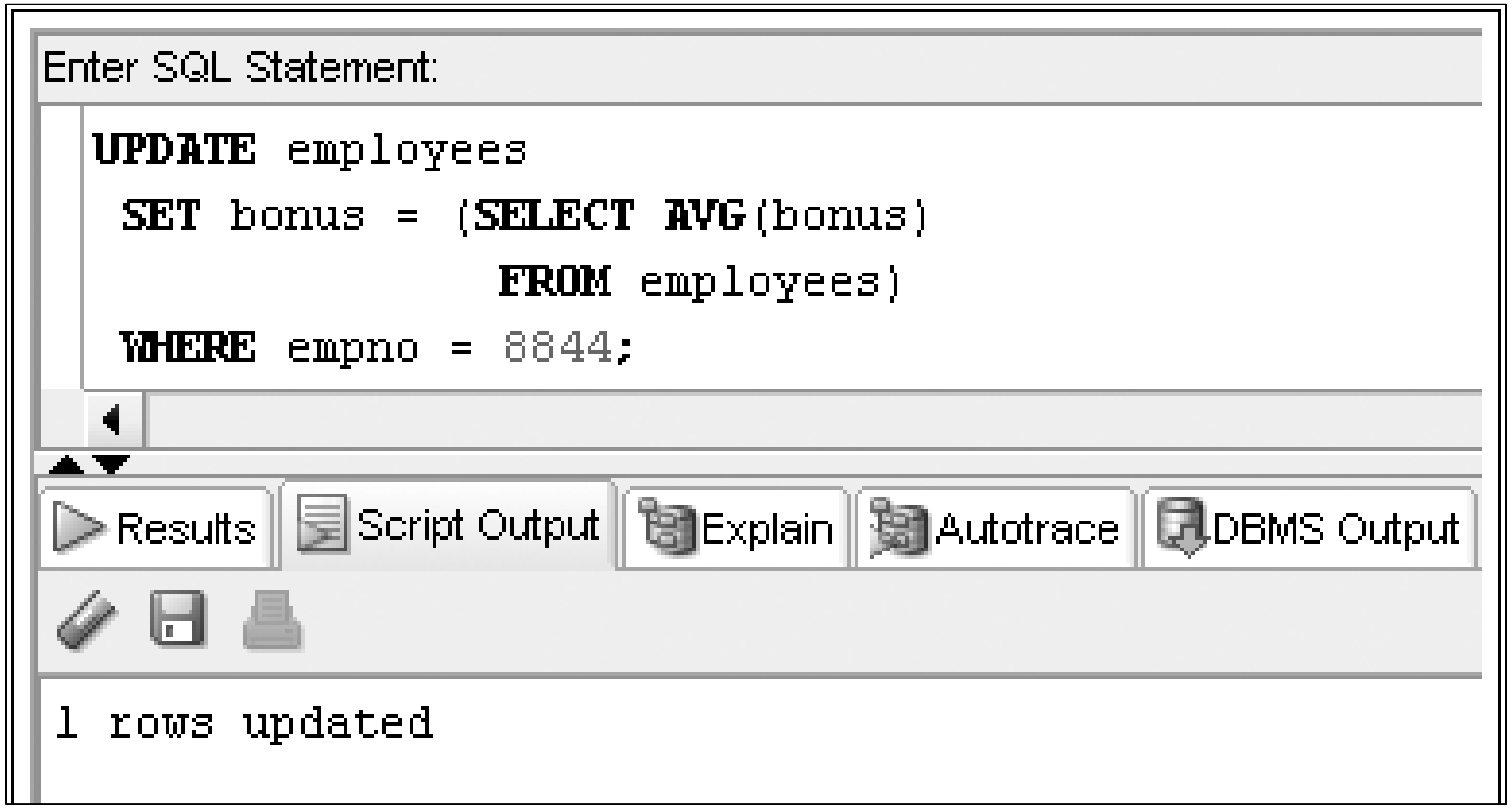
Subquery Factoring Clause…

- XXX

```
WITH dcount AS (
  SELECT deptno, COUNT(*) AS dcount
  FROM   employees
  GROUP BY deptno)
SELECT e.lname Emp_Lastname,
       e.deptno e_dept,
       d1.dcount edept_count,
       m.lname manager_name,
       m.deptno mdept,
       d2.dcount mdept_count
FROM   employees e,
       dcount d1,
       employees m,
       dcount d2
WHERE  e.deptno = d1.deptno
AND    e.mgr = m.empno
AND    m.deptno = d2.deptno
   AND e.mgr = '7839';
```

Subquery in a DML Action…

- xxx



```
UPDATE employees
 SET bonus = (SELECT AVG(bonus)
              FROM employees)
 WHERE empno = 8844;
```

```
1 rows updated
```

KING GRADUATE SCHOOL MONROE COLLEGE

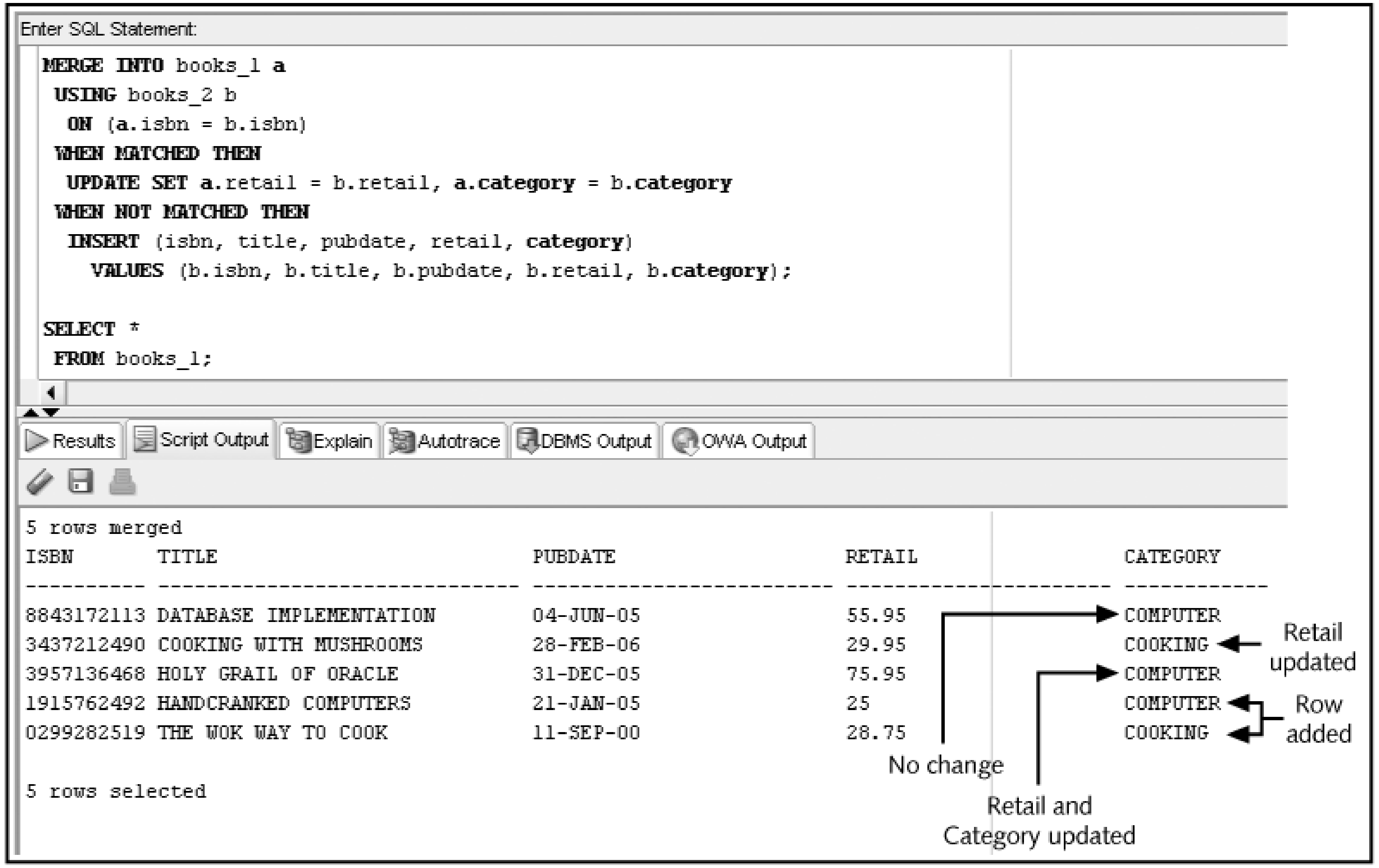
MERGE Statement…

- With a MERGE Statement, a Series of DML Actions Can Occur With a Single SQL Statement…
- Conditionally Updates One Data Source Based On Another…

MERGE Statement – Example – Part 1…

MERGE INTO books_1 a

- The “books_1” Table is to Be Changed and a Table Alias of “a” Is Assigned to This Table…



```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
 WHEN NOT MATCHED THEN
  INSERT (isbn, title, pubdate, retail, category)
    VALUES (b.isbn, b.title, b.pubdate, b.retail, b.category);

SELECT *
 FROM books_1;
```

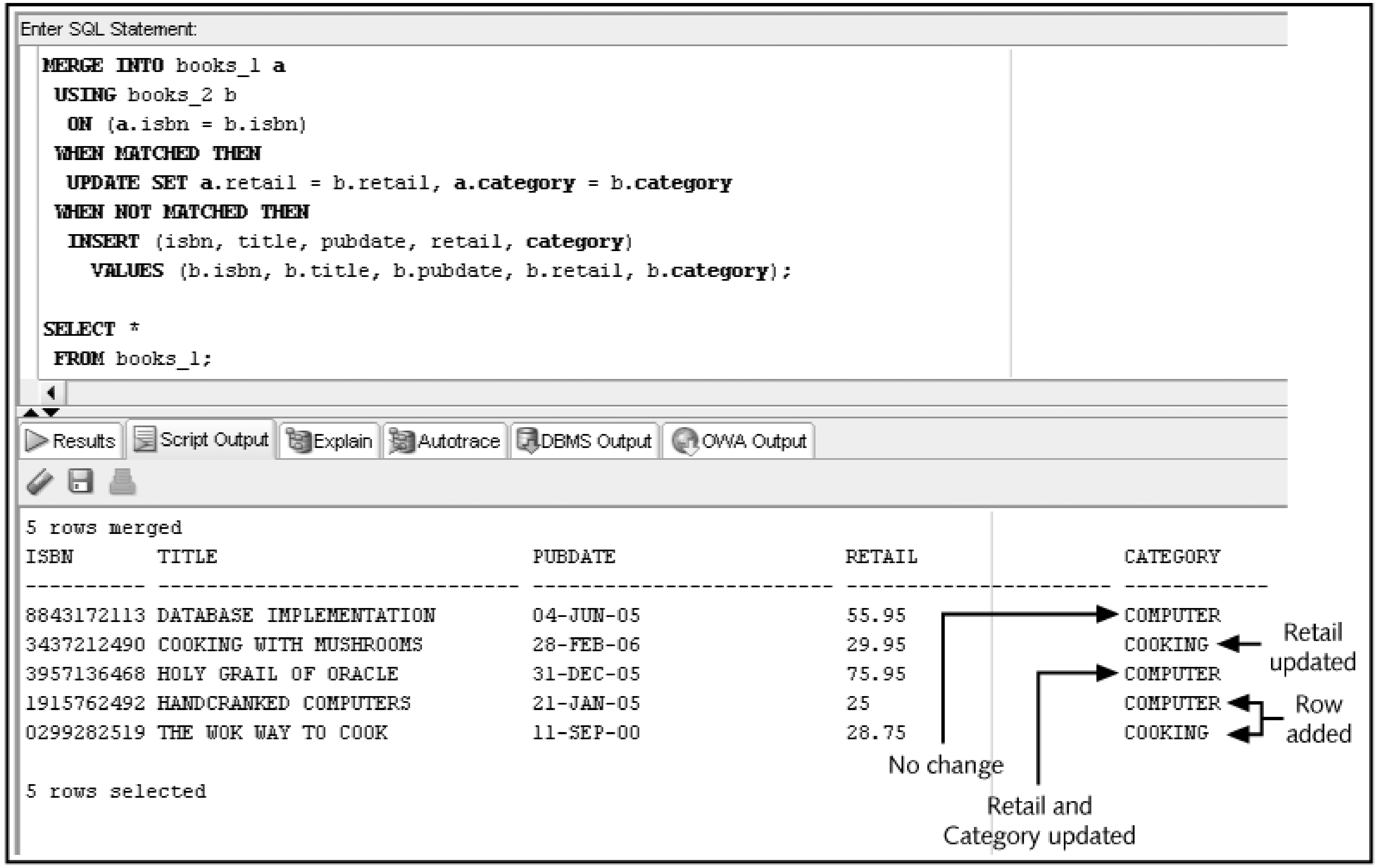
```
5 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL                 CATEGORY
---------- ------------------------------ ------------------------- ---------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                  COMPUTER
3437212490 COOKING WITH MUSHROOMS         28-FEB-06                 29.95                  COOKING
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                  COMPUTER
1915762492 HANDCRANKED COMPUTERS          21-JAN-05                 25                     COMPUTER
0299282519 THE WOK WAY TO COOK            11-SEP-00                 28.75                  COOKING

5 rows selected
```

KING GRADUATE SCHOOL MONROE COLLEGE

MERGE Statement – Example – Part 2…

USING books_2 b

- The “books_2” Table Will Provide Data to Update and/or Insert Into “books_1”…
- A Table Alias of “b” is Assigned to This Table…



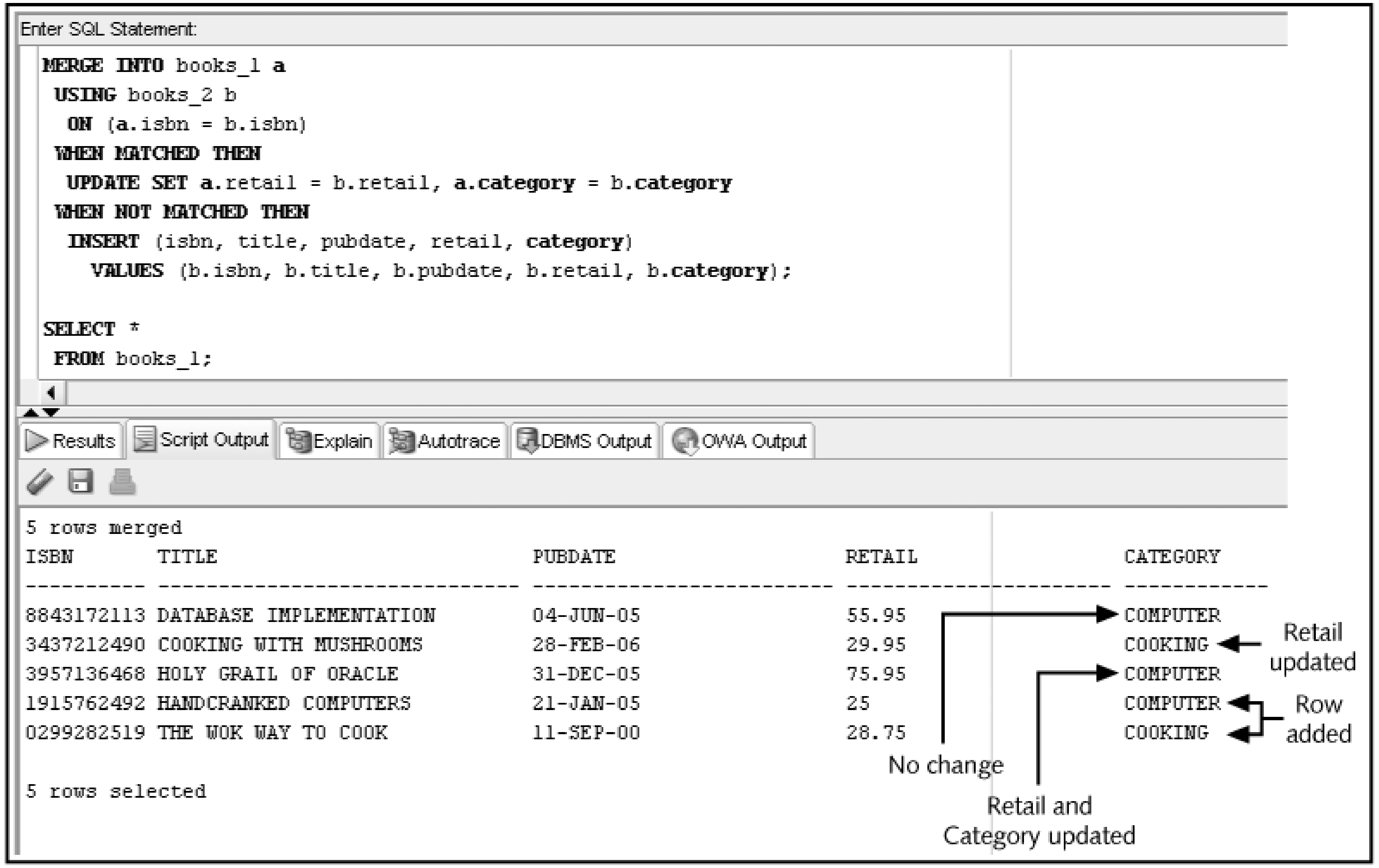
```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
 WHEN NOT MATCHED THEN
  INSERT (isbn, title, pubdate, retail, category)
    VALUES (b.isbn, b.title, b.pubdate, b.retail, b.category);

SELECT *
 FROM books_1;
```

```
5 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL                 CATEGORY
---------- ------------------------------ ------------------------- ---------------------- -----------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                  COMPUTER
3437212490 COOKING WITH MUSHROOMS         28-FEB-06                 29.95                  COOKING
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                  COMPUTER
1915762492 HANDCRANKED COMPUTERS          21-JAN-05                 25                     COMPUTER
0299282519 THE WOK WAY TO COOK            11-SEP-00                 28.75                  COOKING

5 rows selected
```

MERGE Statement – Example – Part 3…

ON (a.isbn = b.isbn)

- Rows of the Two Tables Will Be Joined or Matched Based on “isbn”…



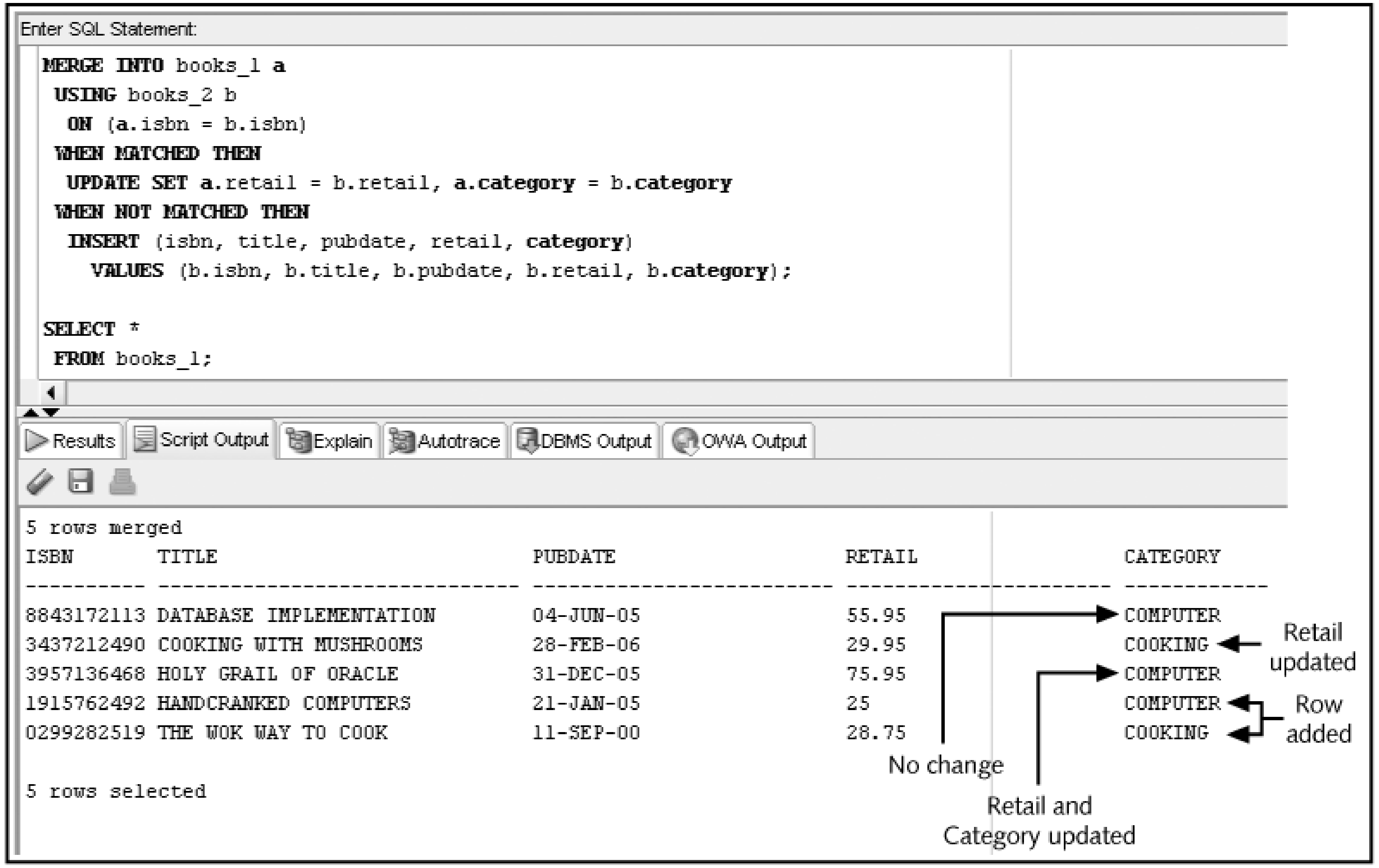
```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
 WHEN NOT MATCHED THEN
  INSERT (isbn, title, pubdate, retail, category)
    VALUES (b.isbn, b.title, b.pubdate, b.retail, b.category);

SELECT *
 FROM books_1;
```

```
5 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL                 CATEGORY
---------- ------------------------------ ------------------------- ---------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                  COMPUTER
3437212490 COOKING WITH MUSHROOMS         28-FEB-06                 29.95                  COOKING
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                  COMPUTER
1915762492 HANDCRANKED COMPUTERS          21-JAN-05                 25                     COMPUTER
0299282519 THE WOK WAY TO COOK            11-SEP-00                 28.75                  COOKING

5 rows selected
```

MERGE Statement – Example – Part 4…

WHEN MATCHED THEN

- If a Row Match Based On "isbn" is Discovered, Execute the "UPDATE" Action in This Clause...
- The "UPDATE" Action Instructs the System to Modify Only Two Columns ("retail" and "category")…



```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
 WHEN NOT MATCHED THEN
  INSERT (isbn, title, pubdate, retail, category)
    VALUES (b.isbn, b.title, b.pubdate, b.retail, b.category);

SELECT *
 FROM books_1;
```

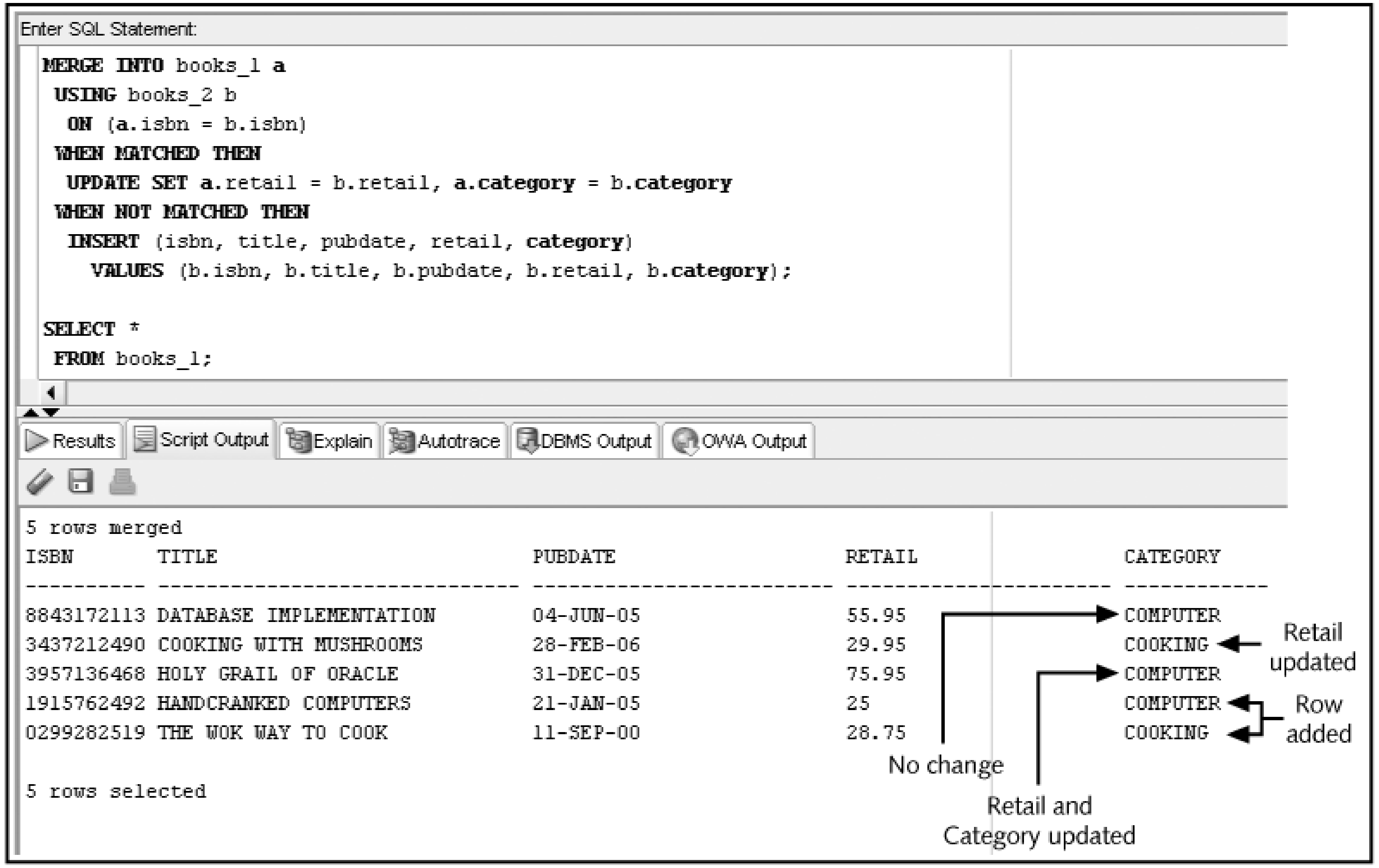
```
5 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL                CATEGORY
---------- ------------------------------ ------------------------- --------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                 COMPUTER
3437212490 COOKING WITH MUSHROOMS         28-FEB-06                 29.95                 COOKING
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                 COMPUTER
1915762492 HANDCRANKED COMPUTERS          21-JAN-05                 25                    COMPUTER
0299282519 THE WOK WAY TO COOK            11-SEP-00                 28.75                 COOKING

5 rows selected
```

KING GRADUATE SCHOOL MONROE COLLEGE

MERGE Statement – Example – Part 5…

WHEN NOT MATCHED THEN

- If No Match is Found Based On The "isbn" (a Book Exists in "books_2" That is Not in "books_1")…
- Then Perform the "INSERT" Action in This Clause…



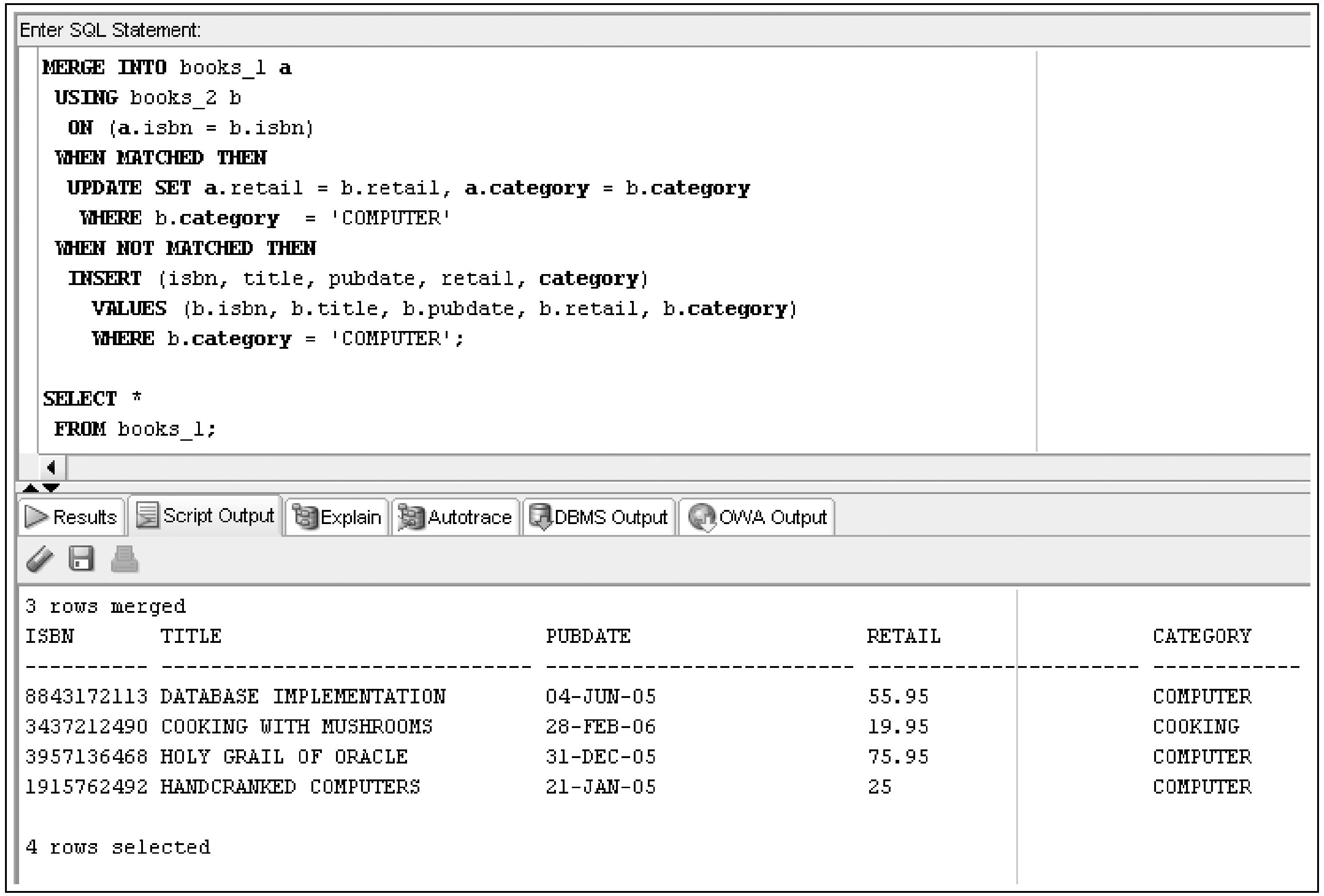
```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
 WHEN NOT MATCHED THEN
  INSERT (isbn, title, pubdate, retail, category)
    VALUES (b.isbn, b.title, b.pubdate, b.retail, b.category);

SELECT *
 FROM books_1;
```

```
5 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL                 CATEGORY
---------- ------------------------------ ------------------------- ---------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                  COMPUTER
3437212490 COOKING WITH MUSHROOMS         28-FEB-06                 29.95                  COOKING
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                  COMPUTER
1915762492 HANDCRANKED COMPUTERS          21-JAN-05                 25                     COMPUTER
0299282519 THE WOK WAY TO COOK            11-SEP-00                 28.75                  COOKING

5 rows selected
```

KING GRADUATE SCHOOL MONROE COLLEGE

MERGE With WHERE Conditions…

- xxx

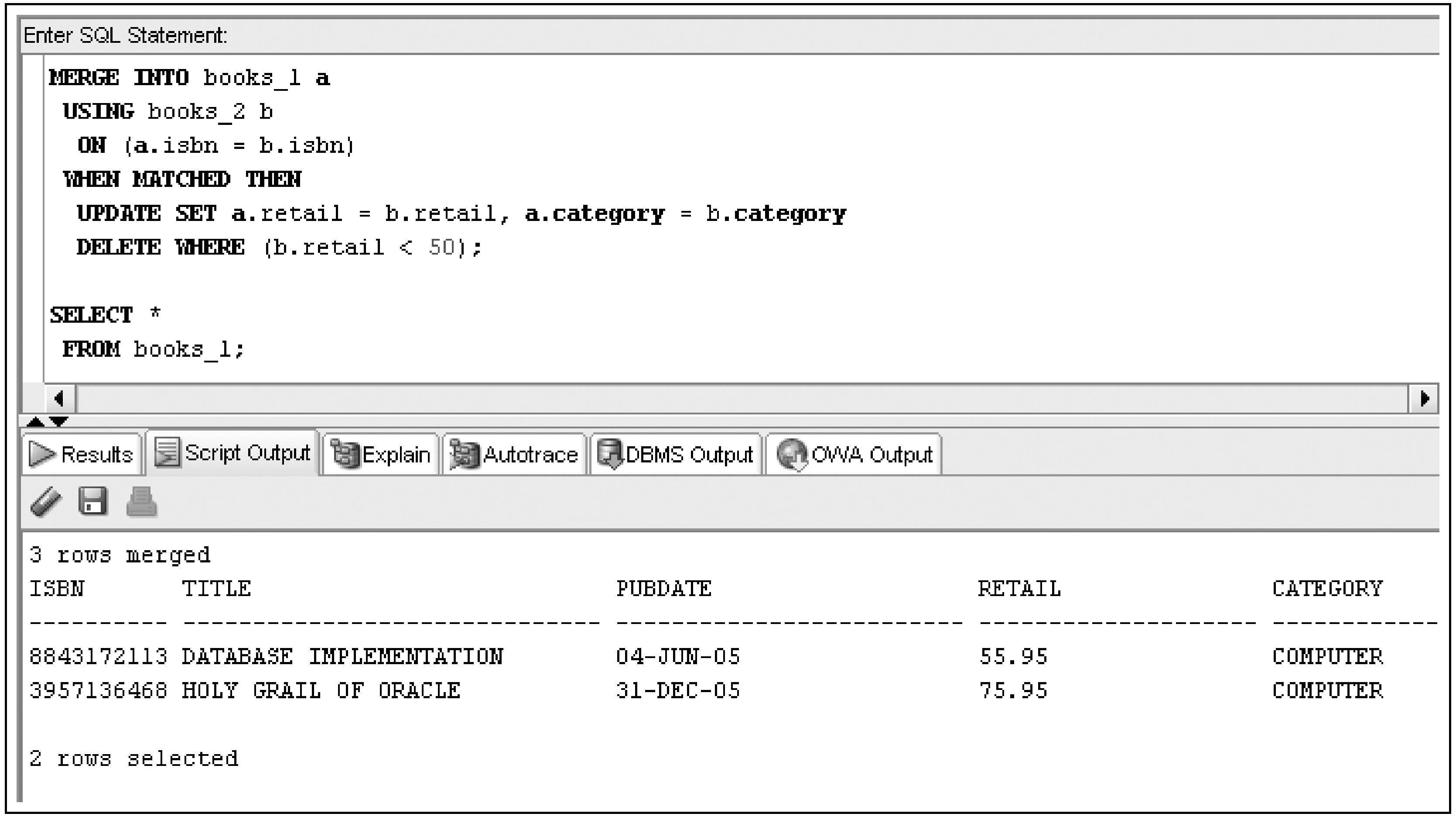


```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
   WHERE b.category  = 'COMPUTER'
 WHEN NOT MATCHED THEN
  INSERT (isbn, title, pubdate, retail, category)
    VALUES (b.isbn, b.title, b.pubdate, b.retail, b.category)
    WHERE b.category = 'COMPUTER';

SELECT *
 FROM books_1;
```

```
3 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL                CATEGORY
---------- ------------------------------ ------------------------- --------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                 COMPUTER
3437212490 COOKING WITH MUSHROOMS         28-FEB-06                 19.95                 COOKING
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                 COMPUTER
1915762492 HANDCRANKED COMPUTERS          21-JAN-05                 25                    COMPUTER

4 rows selected
```

MERGE With DELETE…

- xxx



```
MERGE INTO books_1 a
 USING books_2 b
  ON (a.isbn = b.isbn)
 WHEN MATCHED THEN
  UPDATE SET a.retail = b.retail, a.category = b.category
  DELETE WHERE (b.retail < 50);

SELECT *
 FROM books_1;
```

```
3 rows merged
ISBN       TITLE                          PUBDATE                   RETAIL               CATEGORY
---------- ------------------------------ ------------------------- -------------------- ------------
8843172113 DATABASE IMPLEMENTATION        04-JUN-05                 55.95                COMPUTER
3957136468 HOLY GRAIL OF ORACLE           31-DEC-05                 75.95                COMPUTER

2 rows selected
```

Summary - Part 1…

- A Subquery is a Complete Query Nested in the SELECT, FROM, HAVING, or WHERE Clause of Another Query…
- The Subquery Must Be Enclosed in Parentheses and Have a SELECT and a FROM Clause At a Minimum…
- Subqueries Are Completed First…the Result of the Subquery Is Used As Input For the Outer Query…
- A Single-Row Subquery Can Return a Maximum of One Value…
- Single-Row Operators Include =, >, <, >=, <=, And <>…
- Multiple-Row Subqueries Return More Than One Row of Results…

Summary - Part 2…

- Operators That Can Be Used With Multiple-Row Subqueries Include IN, ALL, ANY, and EXISTS…
- Multiple-Column Subqueries Return More Than One Column to the Outer Query…
- NULL Values Returned By a Multiple-Row or Multiple-Column Subquery Will Not Present a Problem If the “IN” or “=ANY” Operator Is Used…
- Correlated Subqueries Reference a Column Contained in the Outer Query…
- Subqueries Can Be Nested to a Maximum Depth of 255 Subqueries in the WHERE Clause of the Parent Query…

Summary - Part 3…

- With Nested Subqueries, the Innermost Subquery is Executed First…Then the Next Highest-Level Subquery is Executed and So On Until the Outermost Query is Reached…
- A MERGE Statement Allows Multiple DML Actions to Be Conditionally Performed While Comparing Data of Two Tables…

KING GRADUATE SCHOOL MONROE COLLEGE

Questions…