ISTN212

Chapter 6: Procedural Language

SQL and Advanced SQL

SQL Join Operators

- Ability to combine (join) tables on common attributes is most important distinction between relational database and other databases
- Join is performed when data are retrieved from more than one table at a time
- Join is generally composed of an equality comparison between foreign key and primary key of related tables

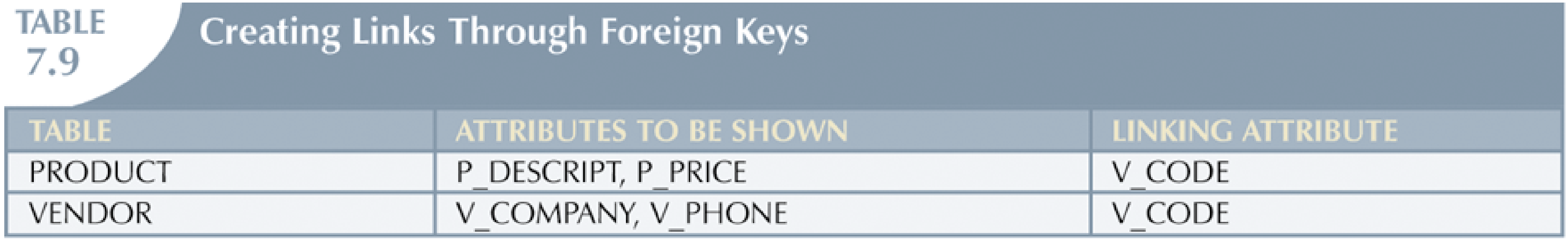


TABLE 7.9 Creating Links Through Foreign Keys

TABLE	ATTRIBUTES TO BE SHOWN	LINKING ATTRIBUTE
PRODUCT	P_DESCRIPT, P_PRICE	V_CODE
VENDOR	V_COMPANY, V_PHONE	V_CODE

JOIN

- Allows us to combine information from two or more tables. Join is the real power behind relational database, allowing the use of independent tables linked by common attributes.

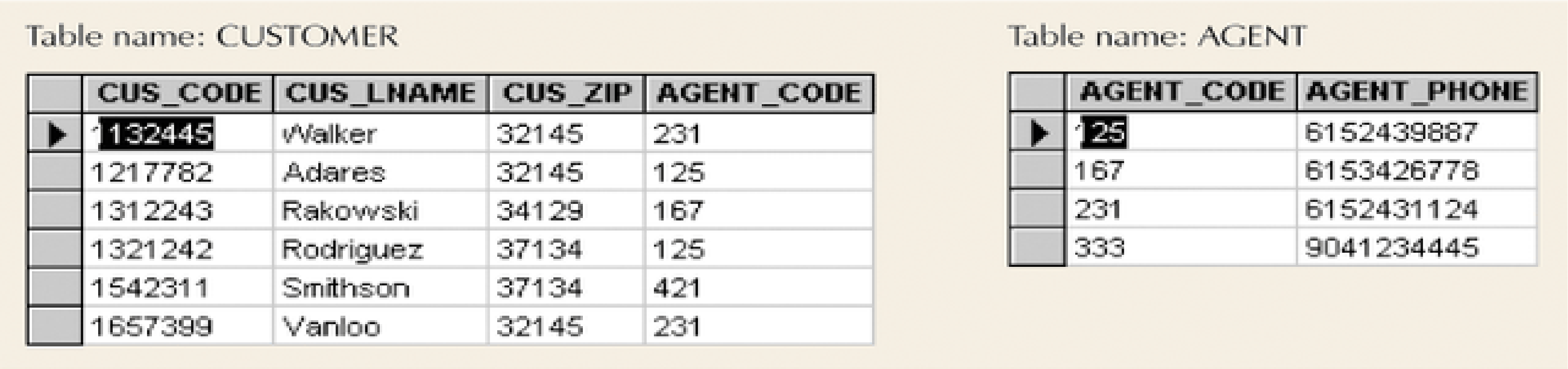


Table name: CUSTOMER

	CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
▶	1132445	Walker	32145	231
	1217782	Adares	32145	125
	1312243	Rakowski	34129	167
	1321242	Rodriguez	37134	125
	1542311	Smithson	37134	421
	1657399	Vanloo	32145	231

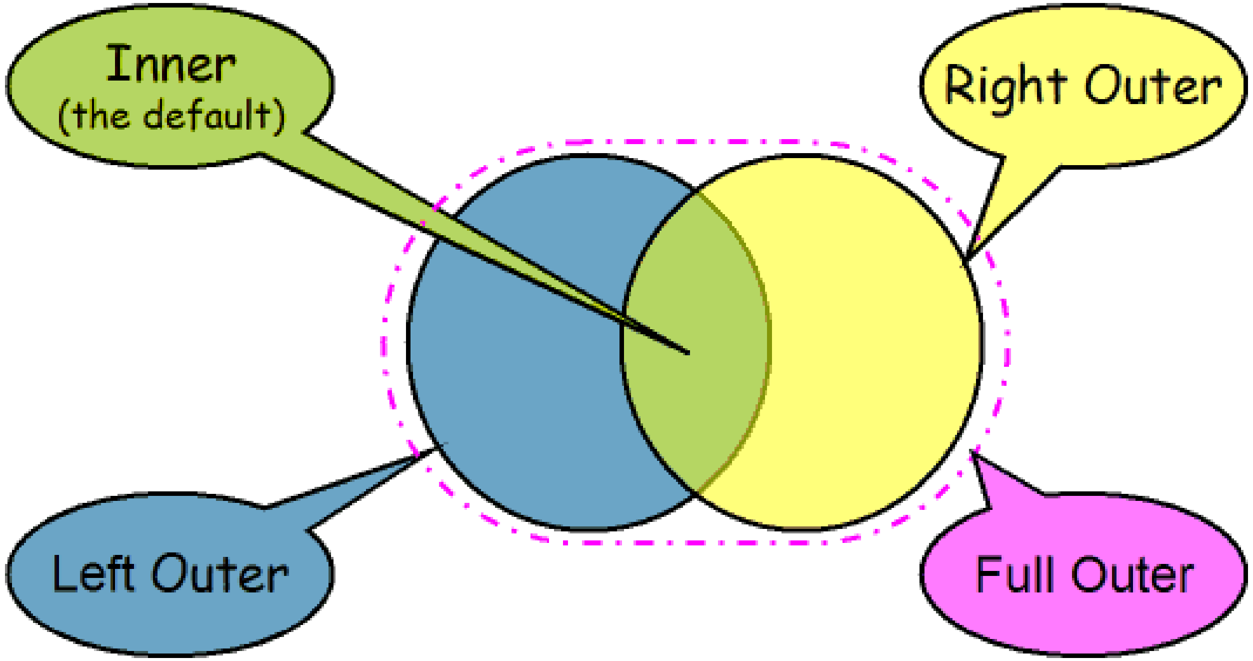
Table name: AGENT

	AGENT_CODE	AGENT_PHONE
▶	125	6152439887
	167	6153426778
	231	6152431124
	333	9041234445

FIGURE 2.11 Two Tables That Will Be Used in JOIN Illustrations

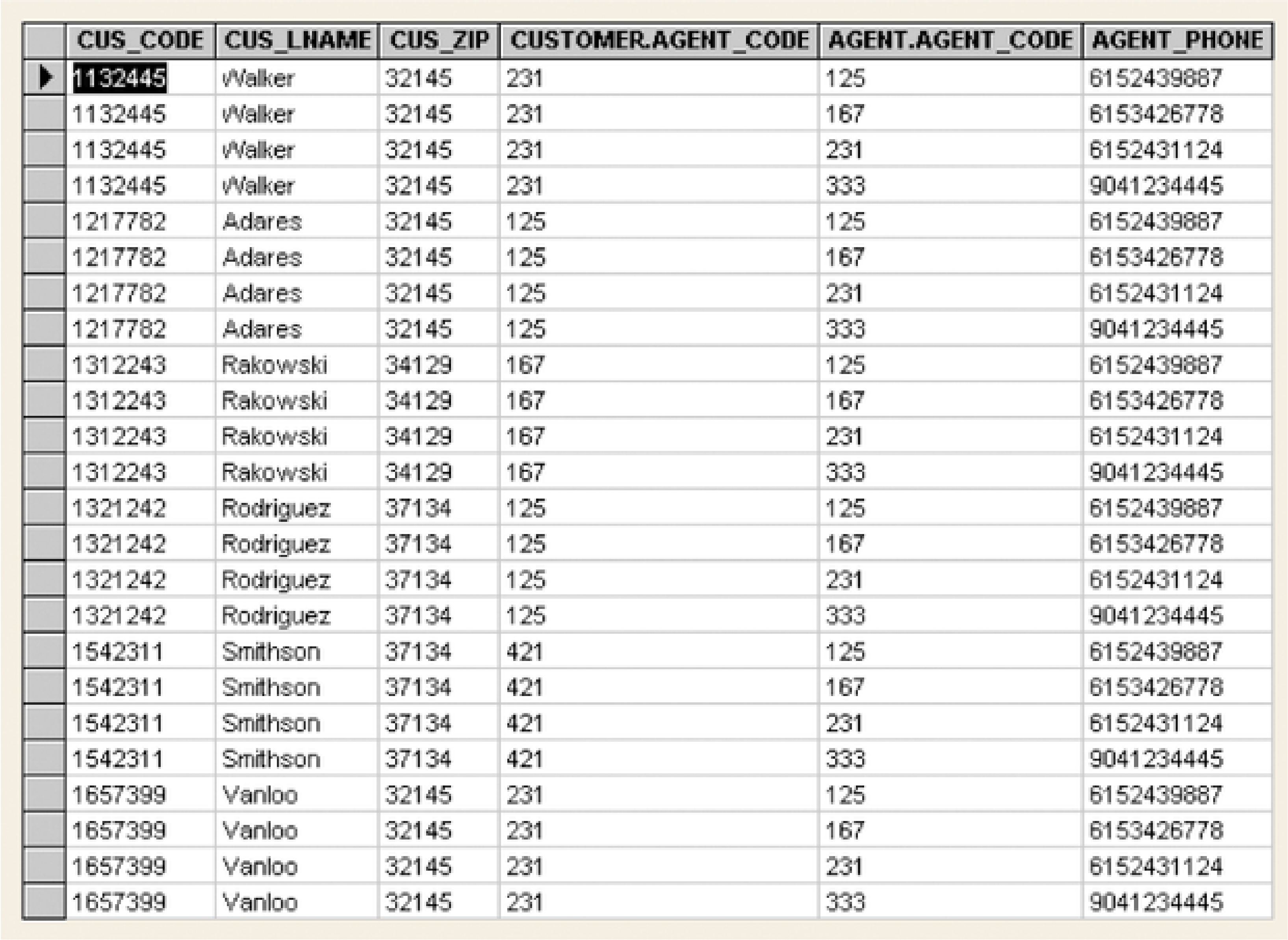
JOIN

- Many types of JOINs



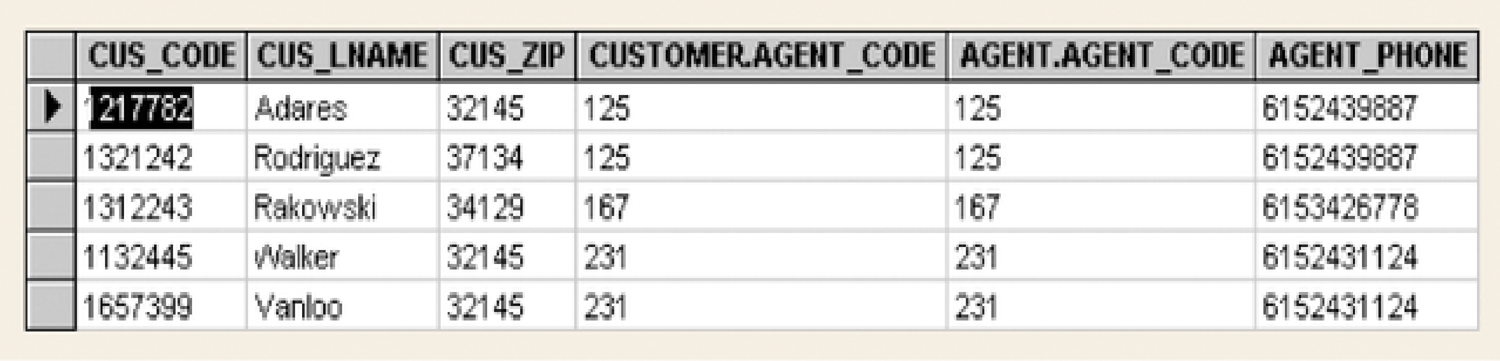
JOIN

- Links tables by selecting rows with common values in common attribute(s)
- Three-stage process
- Creates ONE table
- Stage 1: Apply Product relational operator
- Stage 2: Select yields appropriate rows
- Stage 3: Project removes any duplicate columns



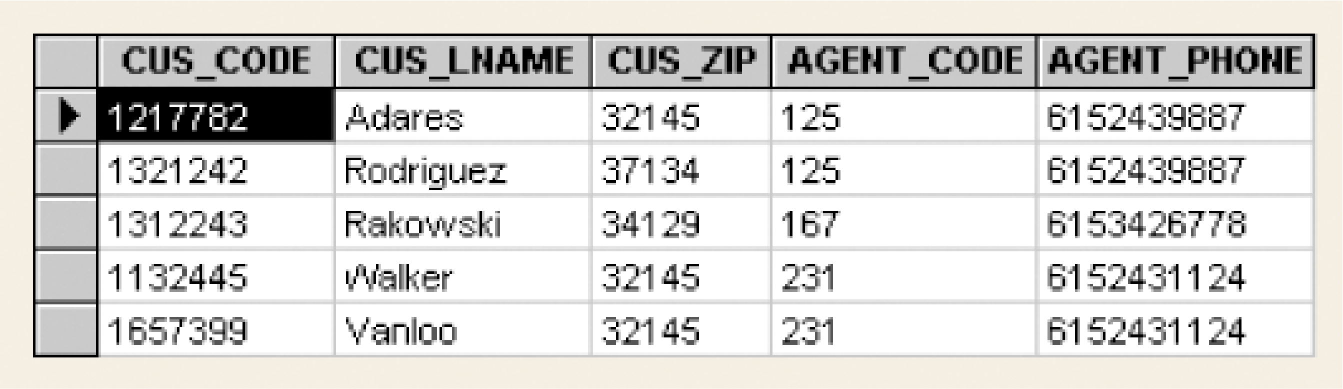
CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1132445	Walker	32145	231	125	6152439887
1132445	Walker	32145	231	167	6153426778
1132445	Walker	32145	231	231	6152431124
1132445	Walker	32145	231	333	9041234445
1217782	Adares	32145	125	125	6152439887
1217782	Adares	32145	125	167	6153426778
1217782	Adares	32145	125	231	6152431124
1217782	Adares	32145	125	333	9041234445
1312243	Rakowski	34129	167	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1312243	Rakowski	34129	167	231	6152431124
1312243	Rakowski	34129	167	333	9041234445
1321242	Rodriguez	37134	125	125	6152439887
1321242	Rodriguez	37134	125	167	6153426778
1321242	Rodriguez	37134	125	231	6152431124
1321242	Rodriguez	37134	125	333	9041234445
1542311	Smithson	37134	421	125	6152439887
1542311	Smithson	37134	421	167	6153426778
1542311	Smithson	37134	421	231	6152431124
1542311	Smithson	37134	421	333	9041234445
1657399	Vanloo	32145	231	125	6152439887
1657399	Vanloo	32145	231	167	6153426778
1657399	Vanloo	32145	231	231	6152431124
1657399	Vanloo	32145	231	333	9041234445

FIGURE 2.12 Natural Join, Step 1: Product



CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124

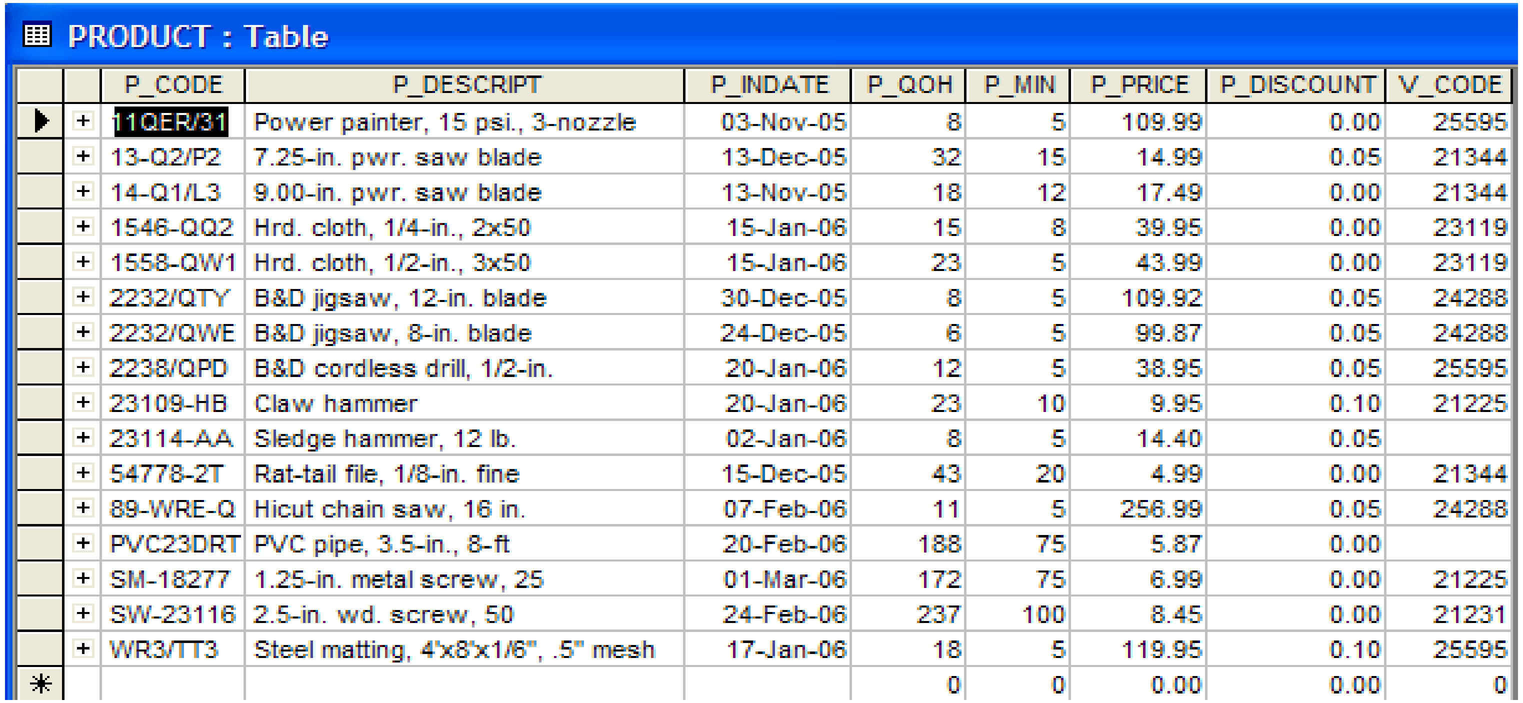
FIGURE 2.13 Natural Join, Step 2: Select



CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

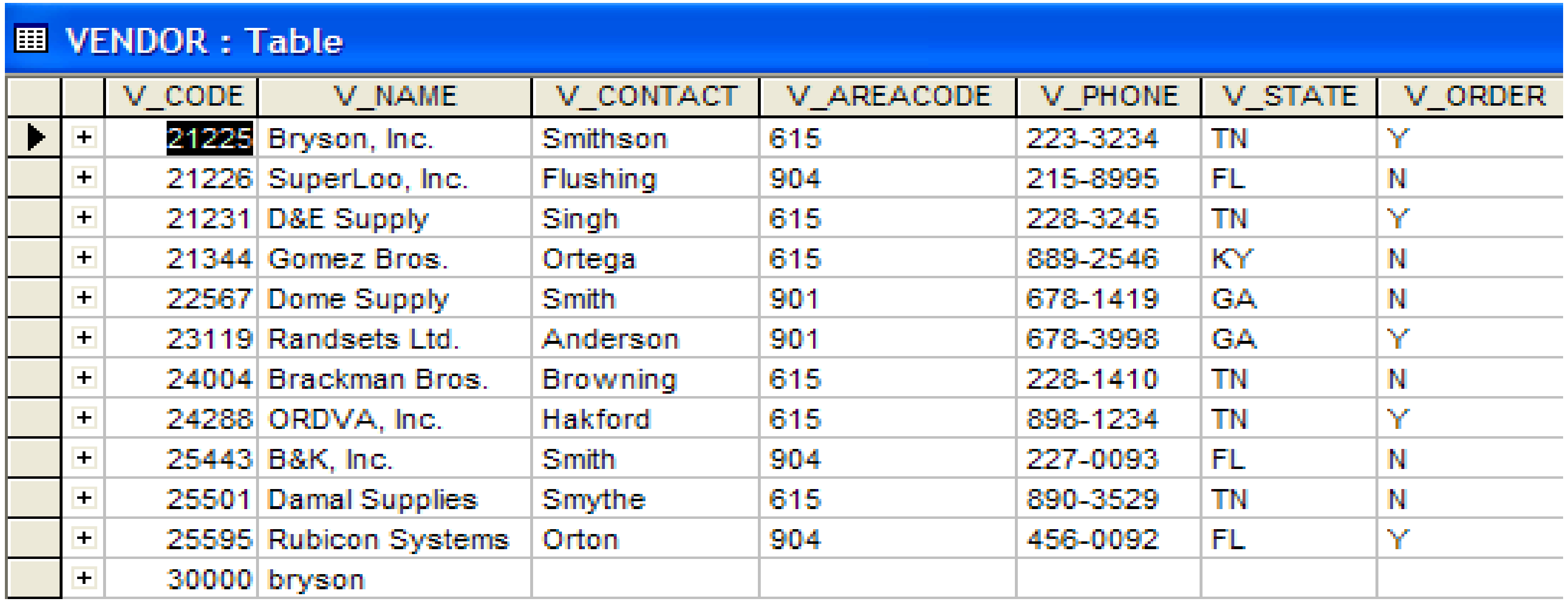
FIGURE 2.14 Natural Join, Step 3: Project

Example 2



PRODUCT : Table

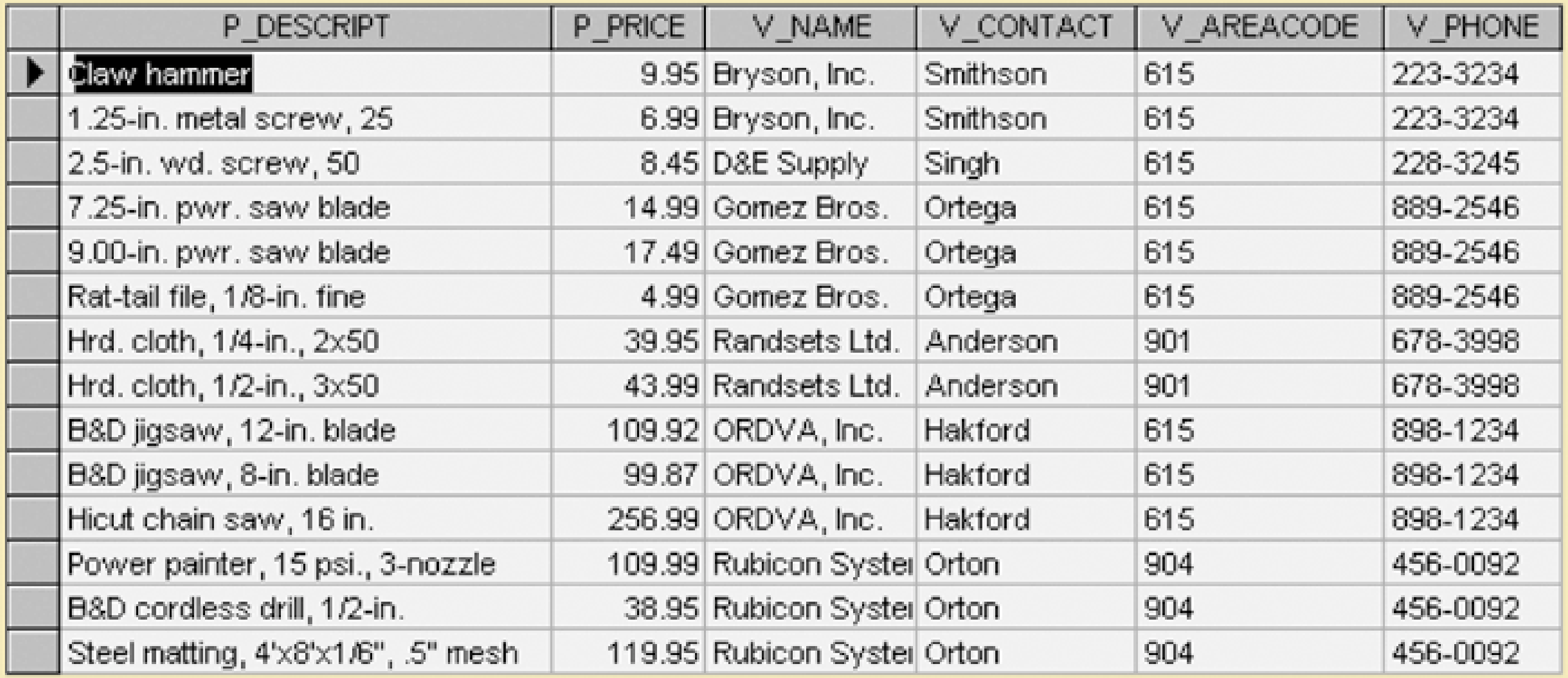
P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-05	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-05	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-05	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-06	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-06	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-05	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-05	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-06	12	5	38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-06	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-06	8	5	14.40	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-05	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-06	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-06	188	75	5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-06	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-06	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-06	18	5	119.95	0.10	25595
			0	0	0.00	0.00	0



VENDOR : Table

V_CODE	V_NAME	V_CONTACT	V_AREACODE	V_PHONE	V_STATE	V_ORDER
21225	Bryson, Inc.	Smithson	615	223-3234	TN	Y
21226	SuperLoo, Inc.	Flushing	904	215-8995	FL	N
21231	D&E Supply	Singh	615	228-3245	TN	Y
21344	Gomez Bros.	Ortega	615	889-2546	KY	N
22567	Dome Supply	Smith	901	678-1419	GA	N
23119	Randsets Ltd.	Anderson	901	678-3998	GA	Y
24004	Brackman Bros.	Browning	615	228-1410	TN	N
24288	ORDVA, Inc.	Hakford	615	898-1234	TN	Y
25443	B&K, Inc.	Smith	904	227-0093	FL	N
25501	Damal Supplies	Smythe	615	890-3529	TN	N
25595	Rubicon Systems	Orton	904	456-0092	FL	Y
30000	bryson					

FIGURE 7.29 The results of a join



P_DESCRIPT	P_PRICE	V_NAME	V_CONTACT	V_AREACODE	V_PHONE
Claw hammer	9.95	Bryson, Inc.	Smithson	615	223-3234
1.25-in. metal screw, 25	6.99	Bryson, Inc.	Smithson	615	223-3234
2.5-in. wd. screw, 50	8.45	D&E Supply	Singh	615	228-3245
7.25-in. pwr. saw blade	14.99	Gomez Bros.	Ortega	615	889-2546
9.00-in. pwr. saw blade	17.49	Gomez Bros.	Ortega	615	889-2546
Rat-tail file, 1/8-in. fine	4.99	Gomez Bros.	Ortega	615	889-2546
Hrd. cloth, 1/4-in., 2x50	39.95	Randsets Ltd.	Anderson	901	678-3998
Hrd. cloth, 1/2-in., 3x50	43.99	Randsets Ltd.	Anderson	901	678-3998
B&D jigsaw, 12-in. blade	109.92	ORDVA, Inc.	Hakford	615	898-1234
B&D jigsaw, 8-in. blade	99.87	ORDVA, Inc.	Hakford	615	898-1234
Hicut chain saw, 16 in.	256.99	ORDVA, Inc.	Hakford	615	898-1234
Power painter, 15 psi., 3-nozzle	109.99	Rubicon Systei	Orton	904	456-0092
B&D cordless drill, 1/2-in.	38.95	Rubicon Systei	Orton	904	456-0092
Steel matting, 4'x8'x1/6", .5" mesh	119.95	Rubicon Systei	Orton	904	456-0092

```
SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT,
   V_AREACODE, V_PHONE
FROM PRODUCT, VENDOR
WHERE PRODUCT.V_CODE = VENDOR.V_CODE
ORDER BY P_PRICE;
```

Will order by Price

JOINing more than two tables

- You need to specify a join condition for each pair of tables
- Number of join will always be N-1, where N represents the number of tables in the from clause
- Eg. Have three tables, you have 2 join clause

```
SELECT CUS_LNAME, INV_NUMBER, INV_DATE, P_DESCRIPT
FROM CUSTOMER, INVOICE, LINE, PRODUCT
WHERE CUSTOMER.CUS_CODE = INVOICE.CUS_CODE
        AND INVOICE.INV_NUMBER = LINE.INV_NUMBER
        AND LINE.P_CODE = PRODUCT.P_CODE
        AND CUSTOMER.CUS_CODE = 10014
```

JOINing tables with an ALIAS

- Alias can be used to identify source table
- Any legal table name can be used as alias – Reduces typing
- Add alias after table name in FROM clause

FROM *tablename alias*

```
SELECT P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT, V_AREACODE, V_PHONE
FROM PRODUCT P, VENDOR V

WHERE P.V_CODE = V.V_CODE

ORDER BY P_PRICE;
```

Natural JOIN

- Returns all rows with matching values in the matching columns
  - Eliminates duplicate columns
- Used when tables share one or more common attributes with common names
- Natural joins may cause problems if columns are added or renamed.
- Syntax:
  - SELECT column-list FROM table1 NATURAL JOIN table2

SELECT dname, ename FROM dept NATURAL JOIN emp

- This is the same as an equi join on (emp.deptno = dept.deptno)

SELECT INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE

FROM INVOICE NATURAL JOIN LINE NATURAL JOIN PRODUCT;

JOIN USING Clause

- Returns only rows with matching values in the column indicated in the USING clause
- Syntax:

SELECT column-list FROM table1 JOIN table2 USING (common-column)

```
SELECT INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE
FROM INVOICE  JOIN LINE
USING (INV_NUMBER) JOIN PRODUCT USING (P_CODE);
```

JOIN ON Clause

- Used when tables have no common attributes name
- Returns only rows that meet the join condition
  - Typically includes equality comparison expression of two columns
- Syntax:

SELECT column-list FROM table1 JOIN table2 ON join-condition

SELECT INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE

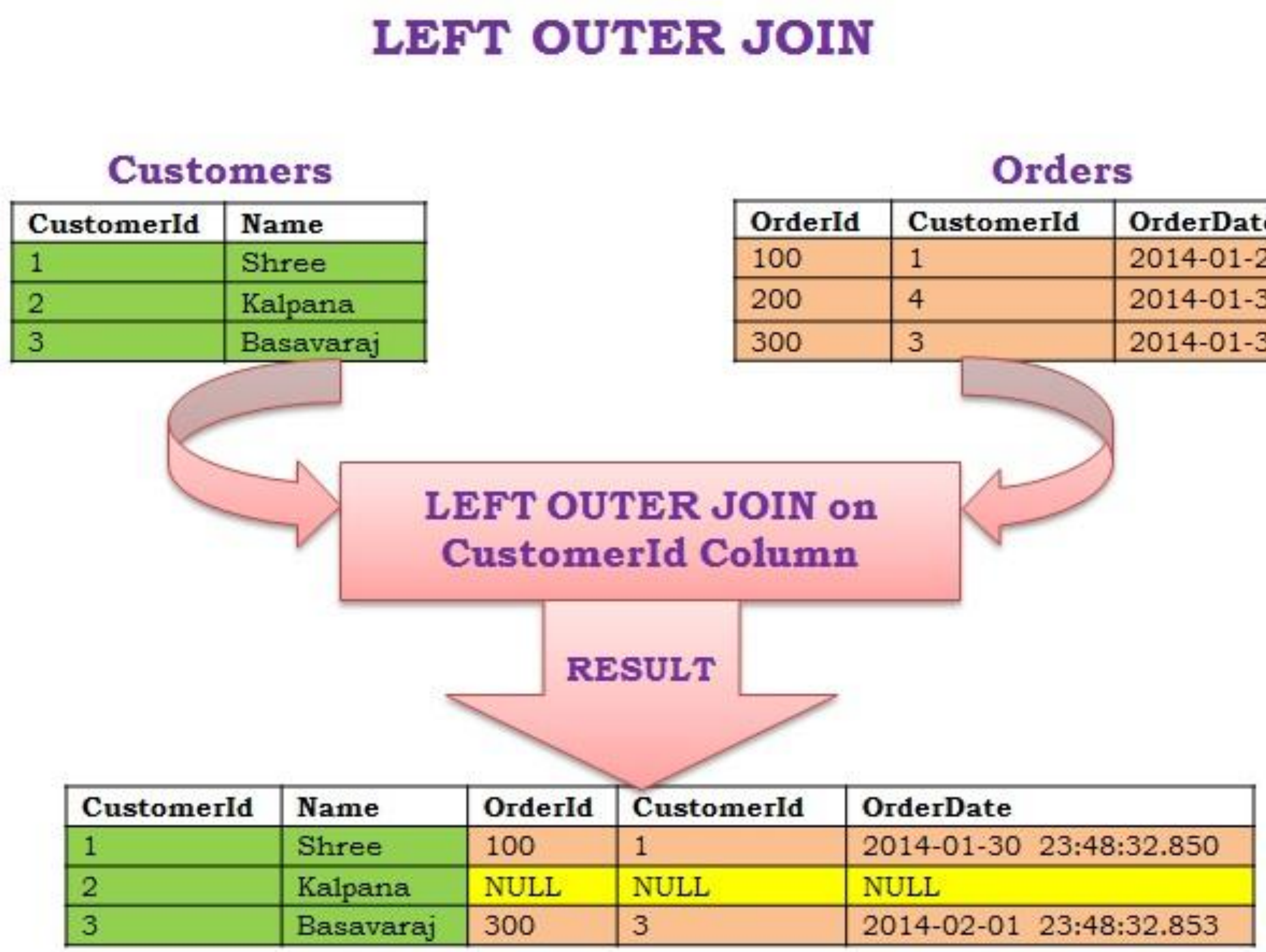
FROM INVOICE JOIN LINE

ON INVOICE.INV_NUMBER = LINE.INV_NUMBER JOIN PRODUCT ON LINE.P_CODE = PRODUCT.P_CODE;

OUTER JOINs

- outer joins returns not only the rows matching the join condition (that is, rows with matching values in the common columns), but also the rows with unmatched values
- 3 types, left, right and full outer
- left outer join will yield not only the rows matching the join condition in the left table, including those that have no matching values in the right table
- in a pair of tables to be joined, a right outer join yields not only the rows matching the join condition in the right table, including the ones with no matching values in the left table.

LEFT OUTER JOIN EXAMPLE



LEFT OUTER JOIN

Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

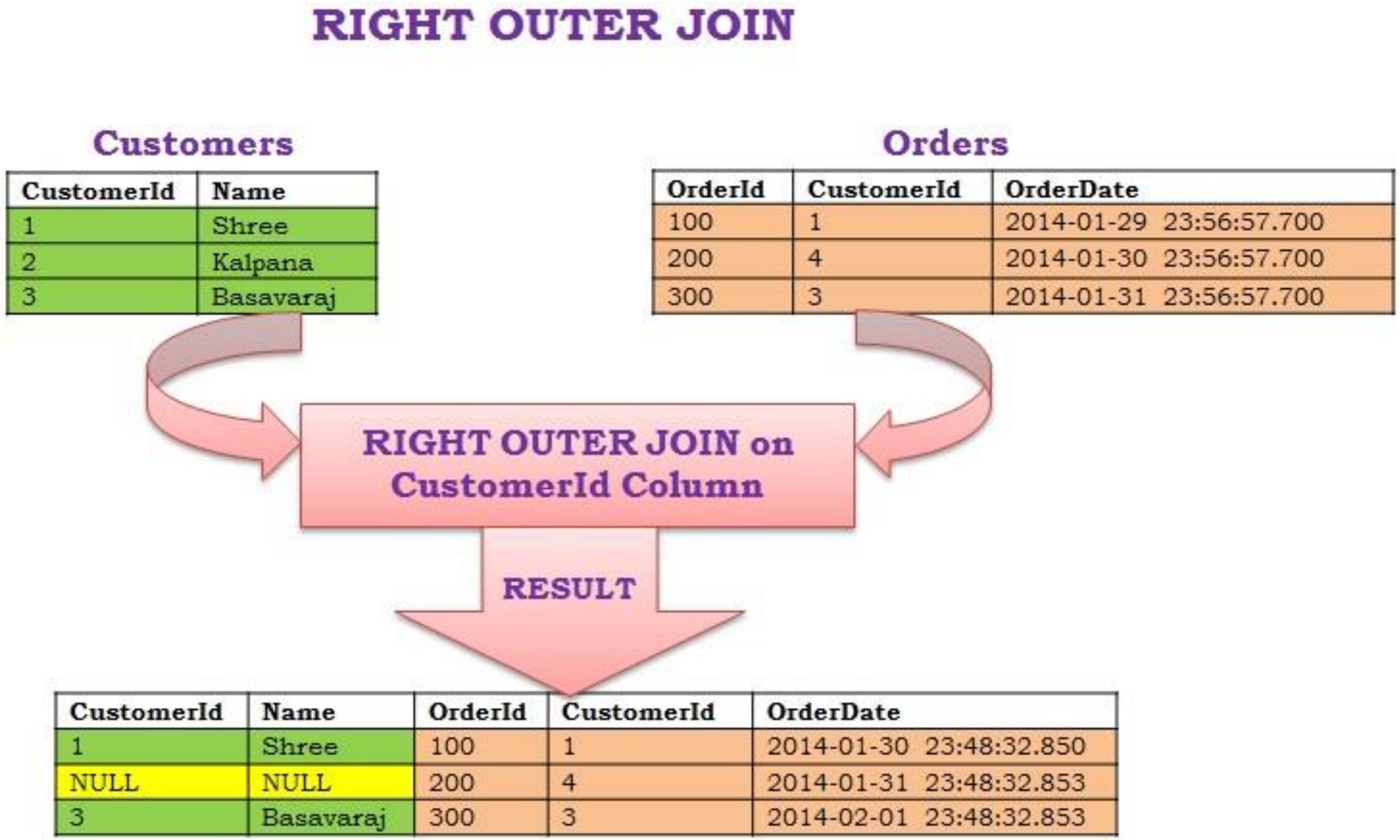
OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

LEFT OUTER JOIN on CustomerId Column

RESULT

CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
2	Kalpana	NULL	NULL	NULL
3	Basavaraj	300	3	2014-02-01 23:48:32.853

RIGHT OUTER JOIN



RIGHT OUTER JOIN

Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

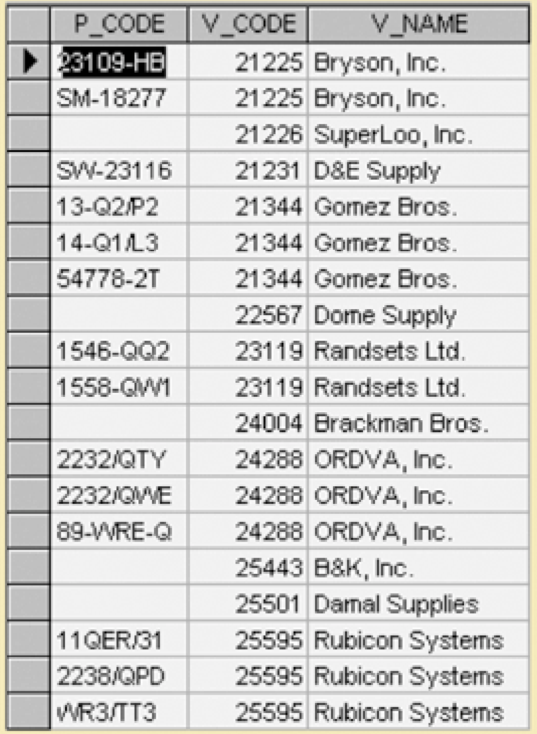
RIGHT OUTER JOIN on CustomerId Column

RESULT

CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
NULL	NULL	200	4	2014-01-31 23:48:32.853
3	Basavaraj	300	3	2014-02-01 23:48:32.853

Outer Joins

FIGURE 7.33 The left outer join results



P_CODE	V_CODE	V_NAME
23109-HB	21225	Bryson, Inc.
SM-18277	21225	Bryson, Inc.
	21226	SuperLoo, Inc.
SW-23116	21231	D&E Supply
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
54778-2T	21344	Gomez Bros.
	22567	Dome Supply
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
	24004	Brackman Bros.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
89-WRE-Q	24288	ORDVA, Inc.
	25443	B&K, Inc.
	25501	Damal Supplies
11QER/31	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
WR3/TT3	25595	Rubicon Systems

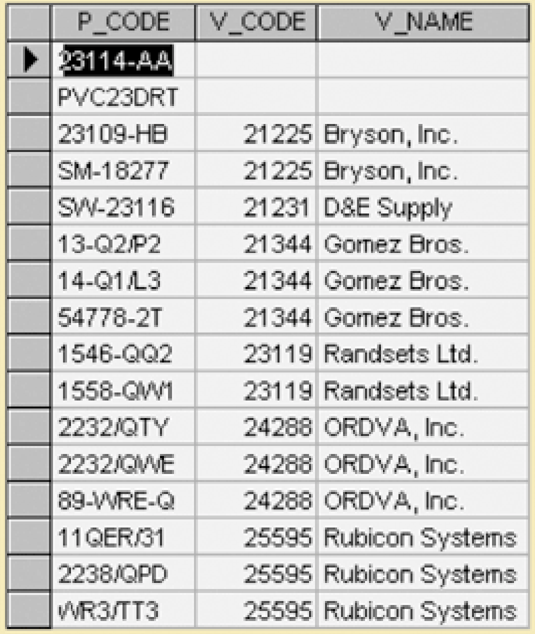
SELECT P_CODE, VENDOR.V_CODE, V_NAME

FROM VENDOR LEFT JOIN PRODUCT ON VENDOR.V_CODE = PRODUCT.V_CODE

Database Principles: Design, Implementation, & Management, 10th Edition, Coronel, Morris and Rob

Outer Joins (continued)

FIGURE 7.34 The right outer join results



P_CODE	V_CODE	V_NAME
23114-AA		
PVC23DRT		
23109-HB	21225	Bryson, Inc.
SM-18277	21225	Bryson, Inc.
SW-23116	21231	D&E Supply
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
54778-2T	21344	Gomez Bros.
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
89-WRE-Q	24288	ORDVA, Inc.
11QER/31	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
WR3/TT3	25595	Rubicon Systems

```
SELECT P_CODE, VENDOR.V_CODE, V_NAME
FROM VENDOR RIGHT JOIN PRODUCT  ON
VENDOR.V_CODE = PRODUCT.V_CODE
```

Database Principles: Design, Implementation, & Management, 10th Edition, Coronel, Morris and Rob

Relational Set Operators

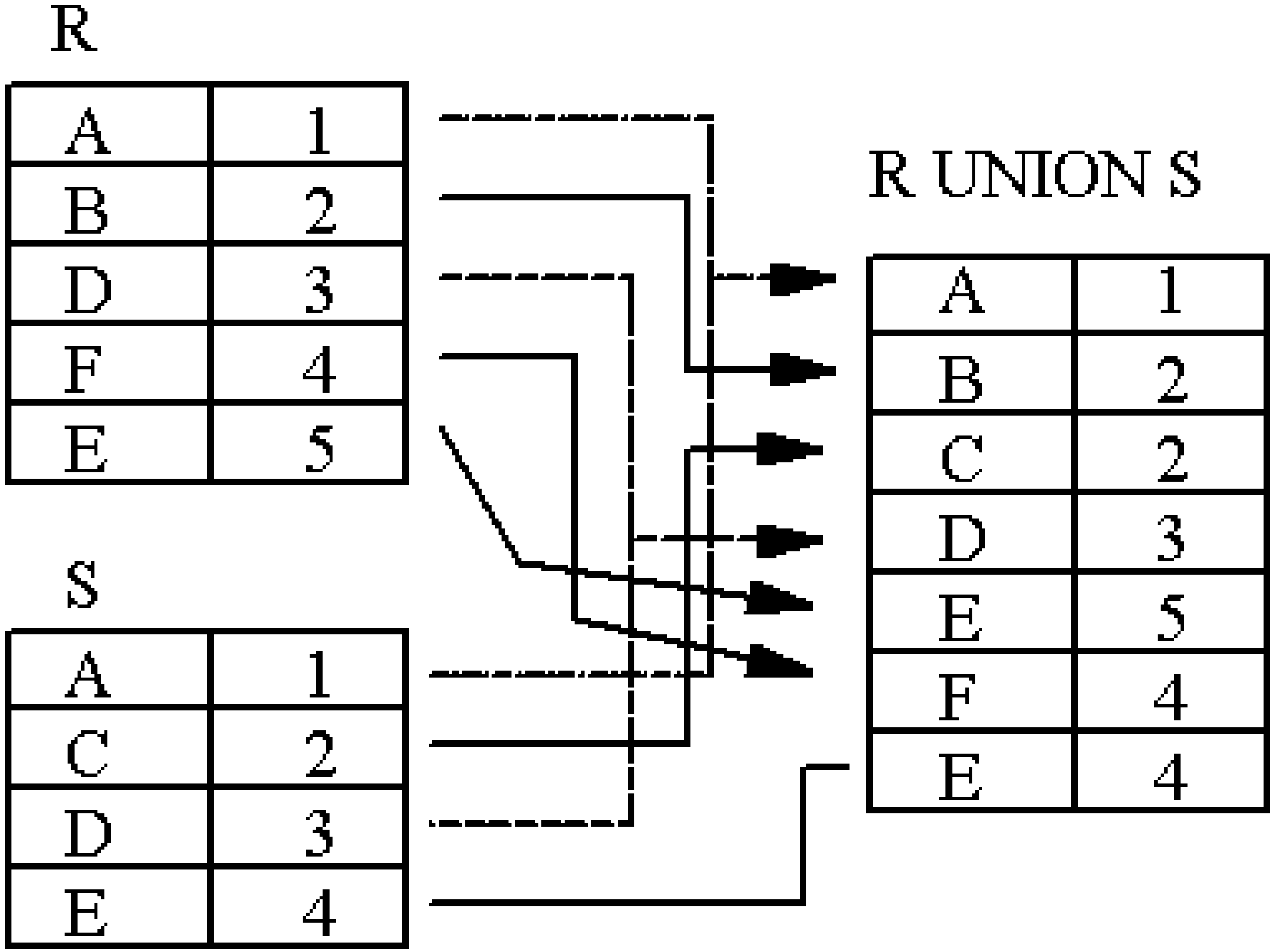
- UNION
- INTERSECT
- MINUS
- Work properly if relations are union-compatible
  - Names of relation attributes must be the same and their data types must be identical

UNION

- Combines rows from two or more queries without including duplicate rows
- Example

```
SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE
FROMCUSTOMER
UNION
SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE
FROM CUSTOMER_2
```

UNION Example



R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R UNION S

A	1
B	2
C	2
D	3
E	5
F	4
E	4

UNION ALL

- Produces a relation that retains duplicate rows
- Example query:

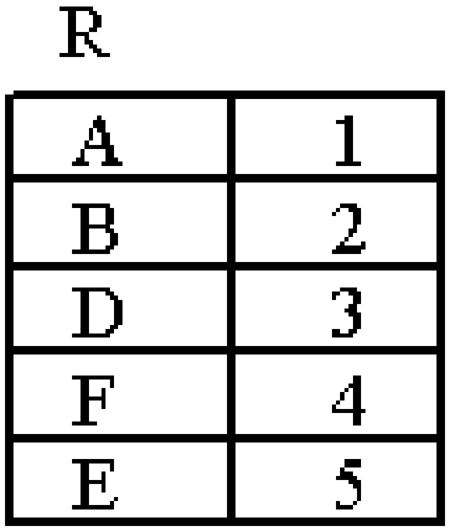
```
SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE
FROMCUSTOMER
UNION ALL
SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE
FROM CUSTOMER_2;
```

INTERSECT

- Combines rows from two queries, returning only the rows that appear in both sets
- Syntax: query INTERSECT query
- Example query:

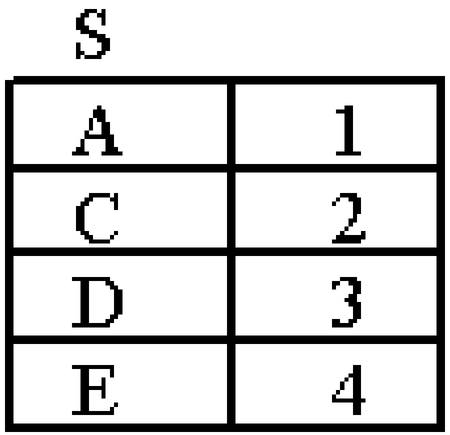
```
SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE
FROM CUSTOMER
INTERSECT
SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE
FROM CUSTOMER_2
```

INTERSECT Example



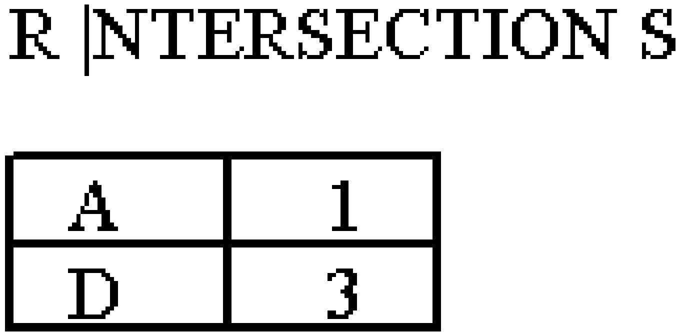
R

A	1
B	2
D	3
F	4
E	5



S

A	1
C	2
D	3
E	4



R |NTERSECTION S

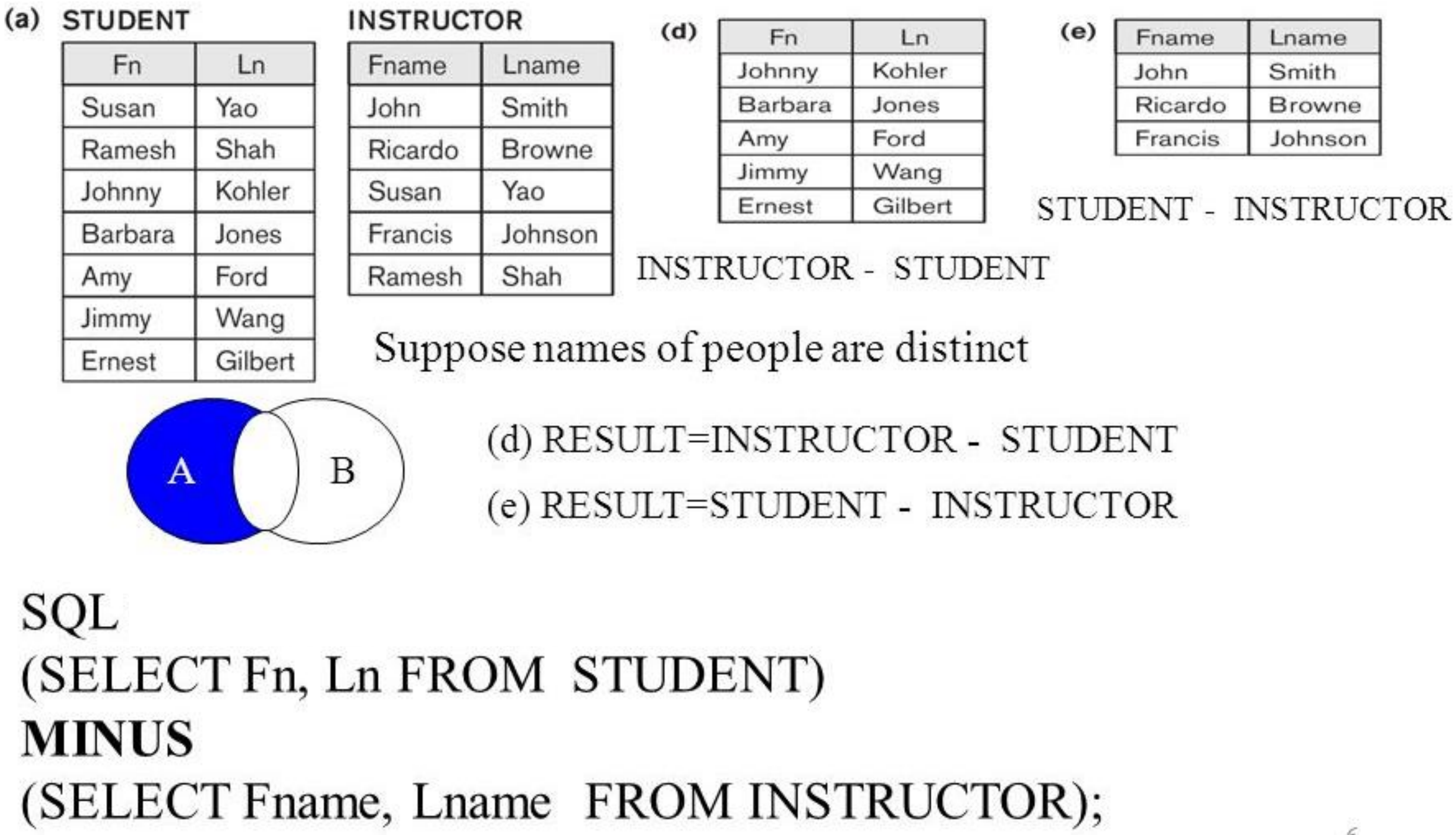
A	1
D	3

MINUS

- Combines rows from two queries
- Returns only the rows that appear in the first set but not in the second
- Syntax: query MINUS query
- Example:

```
SELECT   CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE

FROM CUSTOMER
MINUS

SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE

FROM CUSTOMER_2
```

MINUS Example



(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR - STUDENT

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

STUDENT - INSTRUCTOR

Suppose names of people are distinct

(d) RESULT=INSTRUCTOR - STUDENT

(e) RESULT=STUDENT - INSTRUCTOR

SQL

(SELECT Fn, Ln FROM STUDENT)

MINUS

(SELECT Fname, Lname FROM INSTRUCTOR);

Subqueries

- Often need to process data based on other processed data
  - Eg. Need to generate a list of all products with a price greater than or equal to the average product price
- Is a query inside a query
- A subquery is normally inside parentheses
- The first query in the SQL statement is known as the outer query
- The query inside the SQL statement is known as the inner query
- The inner query is executed first
- The output of an inner query is used as the input for the outer query
- The entire SQL statement is sometimes referred to as a nested query

Subqueries - Examples

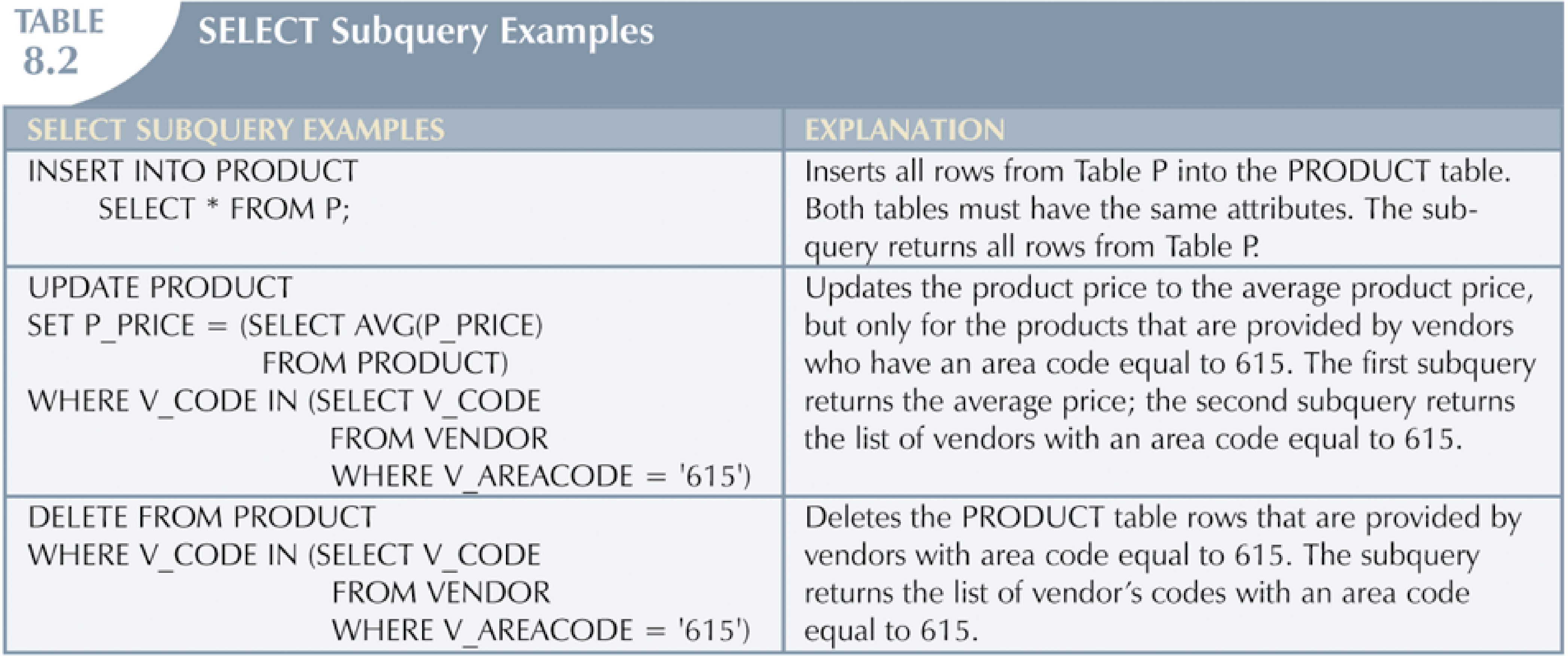


TABLE 8.2 **SELECT Subquery Examples**

SELECT SUBQUERY EXAMPLES	EXPLANATION

INSERT INTO PRODUCT
SELECT * FROM P;
	Inserts all rows from Table P into the PRODUCT table. Both tables must have the same attributes. The subquery returns all rows from Table P.

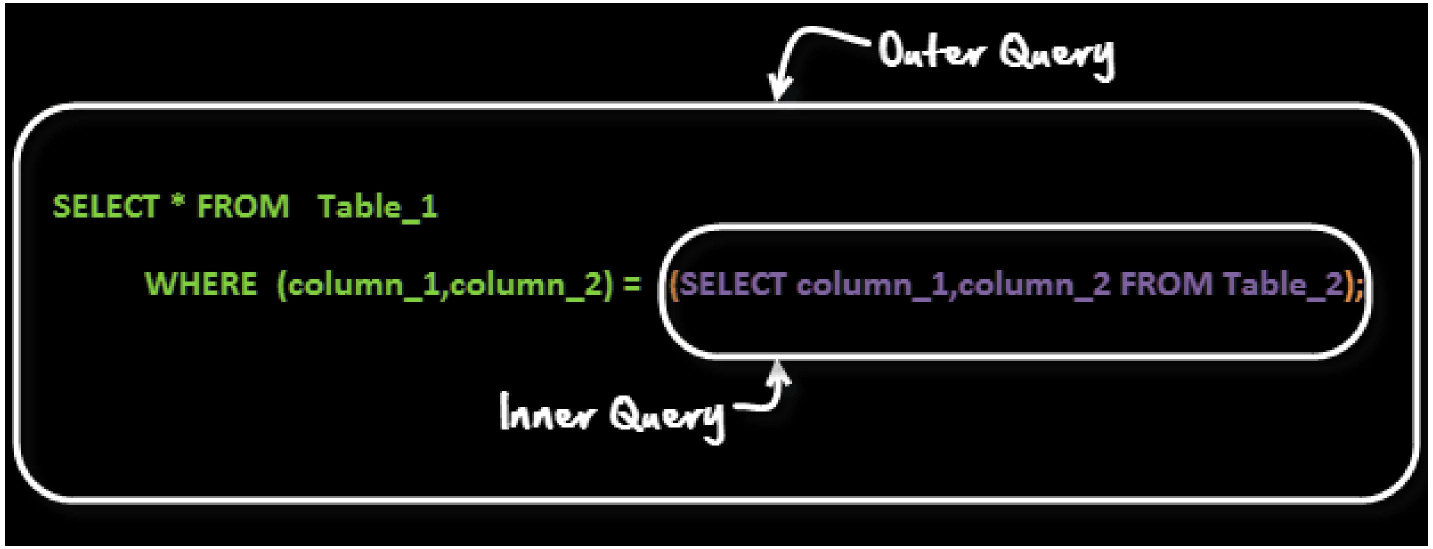
UPDATE PRODUCT
SET P_PRICE = (SELECT AVG(P_PRICE)
FROM PRODUCT)
WHERE V_CODE IN (SELECT V_CODE
FROM VENDOR
WHERE V_AREACODE = '615')
	Updates the product price to the average product price, but only for the products that are provided by vendors who have an area code equal to 615. The first subquery returns the average price; the second subquery returns the list of vendors with an area code equal to 615.

DELETE FROM PRODUCT
WHERE V_CODE IN (SELECT V_CODE
FROM VENDOR
WHERE V_AREACODE = '615')
	Deletes the PRODUCT table rows that are provided by vendors with area code equal to 615. The subquery returns the list of vendor's codes with an area code equal to 615.

Subqueries

- A subquery can return:

- One single value eg. The update example in the previous table
- A list of values (one column and multiple rows) eg. IN
- A virtual table (multicolumn, multirow set of values



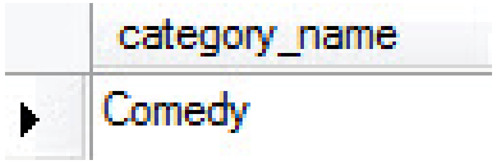
```
SELECT * FROM   Table_1
        WHERE  (column_1,column_2) =  (SELECT column_1,column_2 FROM Table_2);
```

Subquery Example

- A common customer complaint at the MyFlix Video Library is the low number of movie titles. The management wants to buy movies for a category which has least number of titles.

```
SELECT category_name FROM categories WHERE category_id = (SELECT
MIN(category_id) from movies);
```

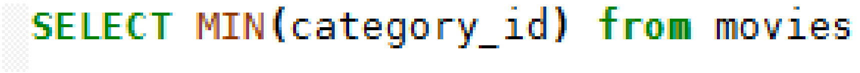
- Result:



category_name
Comedy

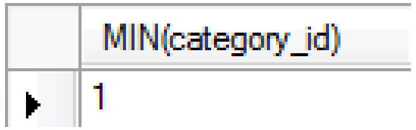
How the subquery works

First the INNER Query is executed



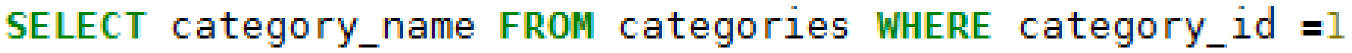
```
SELECT MIN(category_id) from movies
```

INNER Query gives following result



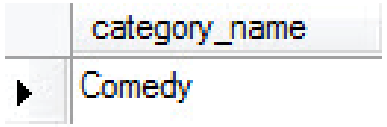
	MIN(category_id)
▶	1

Output of INNER Query is substituted in OUTER Query



```
SELECT category_name FROM categories WHERE category_id =1
```

On Execution OUTER Query gives following Result



	category_name
▶	Comedy

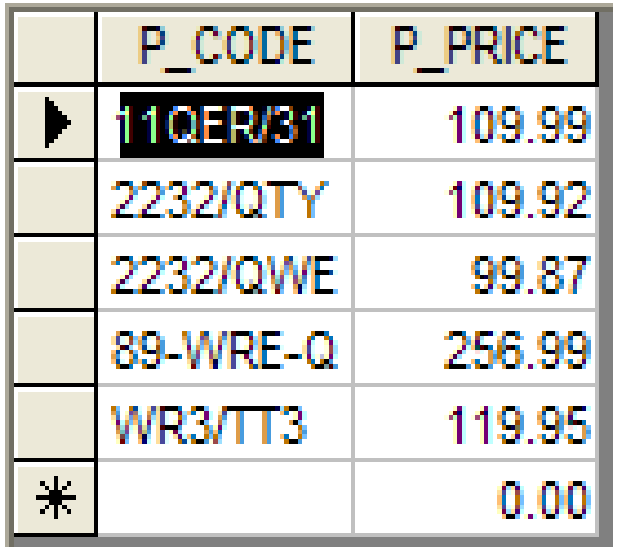
WHERE Subquery

- Uses an inner SELECT subquery on the right side of the WHERE comparison expression
- When used in a >, <, =,>=, or <= conditional expression, requires a subquery that returns only one single value (one column, one row)

```
SELECT P_CODE, P_PRICE
FROM PRODUCT
WHERE P_PRICE >= (SELECT AVG(P_PRICE) FROM PRODUCT);
```

This query done first to get average

Result of first query completes the second query, which is then run



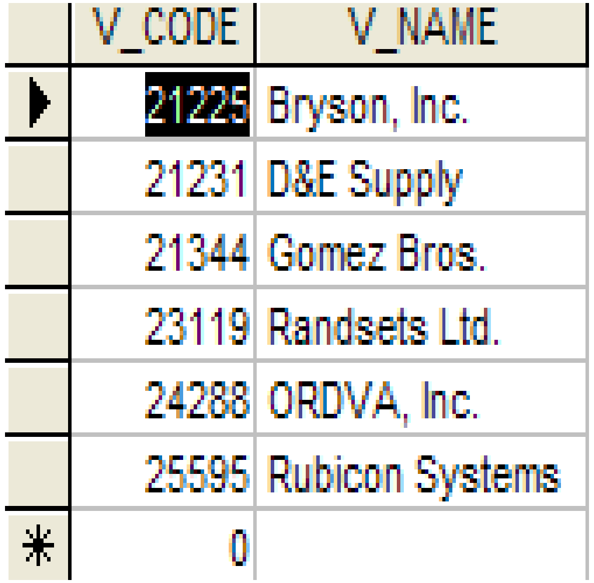
	P_CODE	P_PRICE
▶	11QER/31	109.99
	2232/QTY	109.92
	2232/QWE	99.87
	89-WRE-Q	256.99
	WR3/TT3	119.95
*		0.00

Database Principles: Design, Implementation, & Management, 10th Edition, Coronel, Morris and Rob

IN Subquery

- When you want to compare a single attribute to a list of values, use the IN operator
- When the values are not known beforehand but they can be derived using a query, you must use an IN subquery

```
SELECT V_CODE, V_NAME
FROM VENDOR
WHERE V_CODE IN (SELECT V_CODE FROM PRODUCT);
```

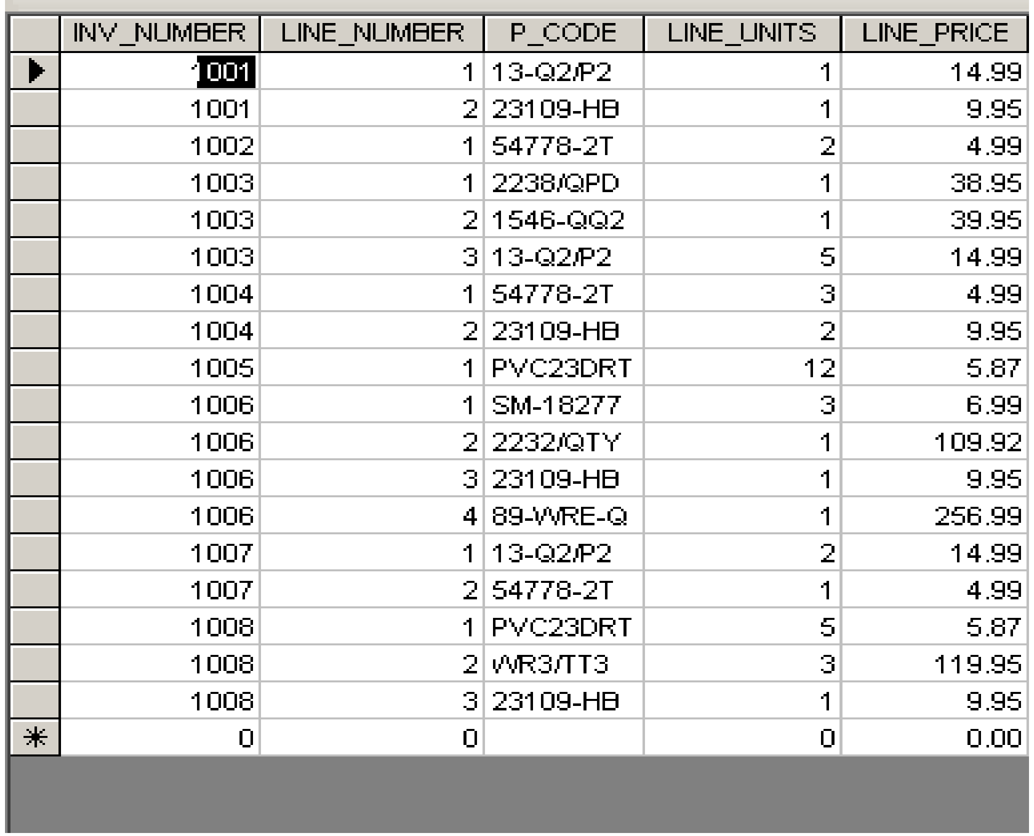


	V_CODE	V_NAME
▶	21225	Bryson, Inc.
	21231	D&E Supply
	21344	Gomez Bros.
	23119	Randsets Ltd.
	24288	ORDVA, Inc.
	25595	Rubicon Systems
*	0	

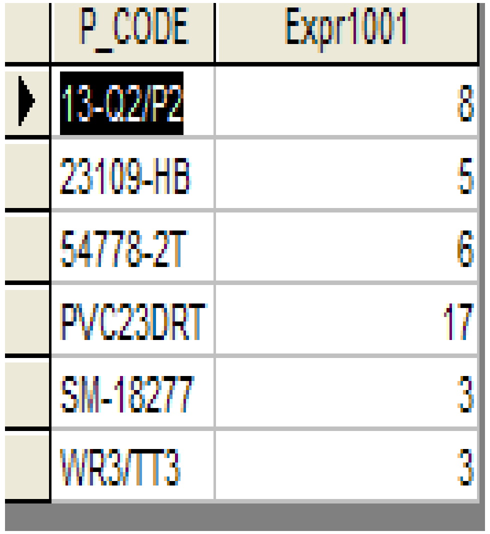
HAVING Subquery

- Just as you can use subqueries with the WHERE clause, you can use a subquery with a HAVING clause
- HAVING clause used to restrict the output of a GROUP BY query

```
SELECT P_CODE, SUM(LINE_UNITS)
FROM LINE
GROUP BY P_CODE
HAVING SUM(LINE_UNITS) > (SELECT AVG(LINE_UNITS) FROM LINE);
```



	INV_NUMBER	LINE_NUMBER	P_CODE	LINE_UNITS	LINE_PRICE
▶	1001	1	13-Q2/P2	1	14.99
	1001	2	23109-HB	1	9.95
	1002	1	54778-2T	2	4.99
	1003	1	2238/QPD	1	38.95
	1003	2	1546-QQ2	1	39.95
	1003	3	13-Q2/P2	5	14.99
	1004	1	54778-2T	3	4.99
	1004	2	23109-HB	2	9.95
	1005	1	PVC23DRT	12	5.87
	1006	1	SM-18277	3	6.99
	1006	2	2232/QTY	1	109.92
	1006	3	23109-HB	1	9.95
	1006	4	89-WRE-Q	1	256.99
	1007	1	13-Q2/P2	2	14.99
	1007	2	54778-2T	1	4.99
	1008	1	PVC23DRT	5	5.87
	1008	2	WR3/TT3	3	119.95
	1008	3	23109-HB	1	9.95
✱	0	0		0	0.00



	P_CODE	Expr1001
▶	13-Q2/P2	8
	23109-HB	5
	54778-2T	6
	PVC23DRT	17
	SM-18277	3
	WR3/TT3	3

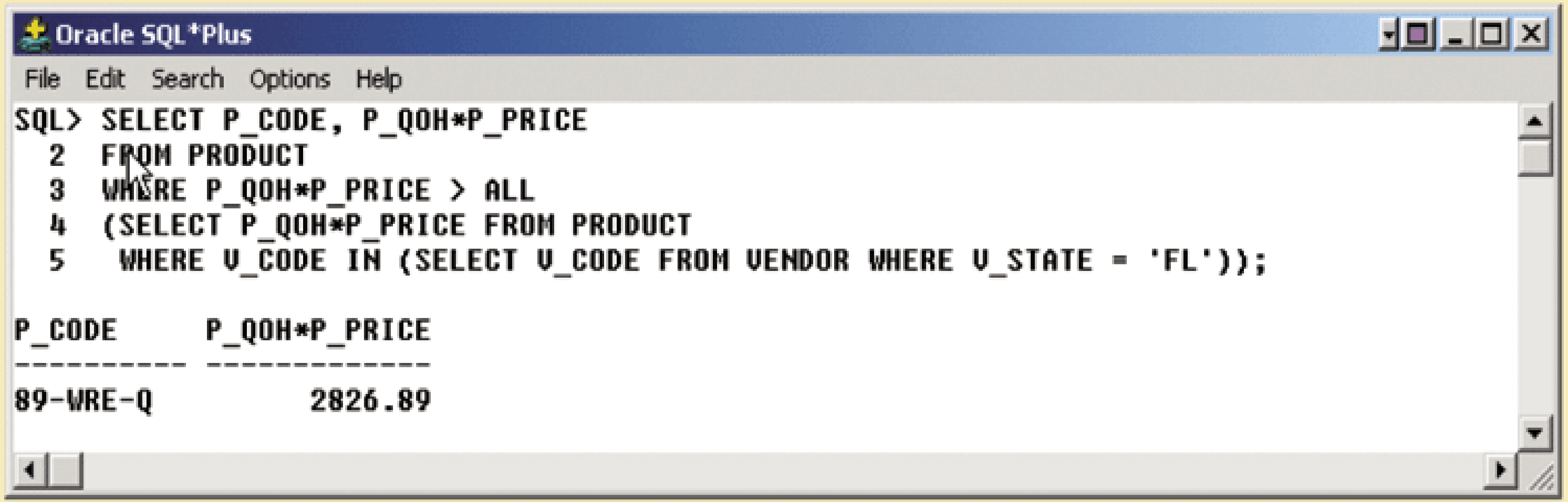
Database Principles: Design, Implementation, & Management, 10th Edition, Coronel, Morris and Rob

Multirow subquery operators – ANY and ALL

- IN subquery used when you need to compare a value to a list of values
- However, IN subquery uses an equality operator (ie. it selects only those rows that match at least one of the values in the list
- For inequality comparison (> or <) of one value to a list of values (use ALL)
- EG. Suppose you want to know what products have a cost that is greater than all individual product costs for products provided by vendors from Florida

Multirow Subquery Operators: ANY and ALL

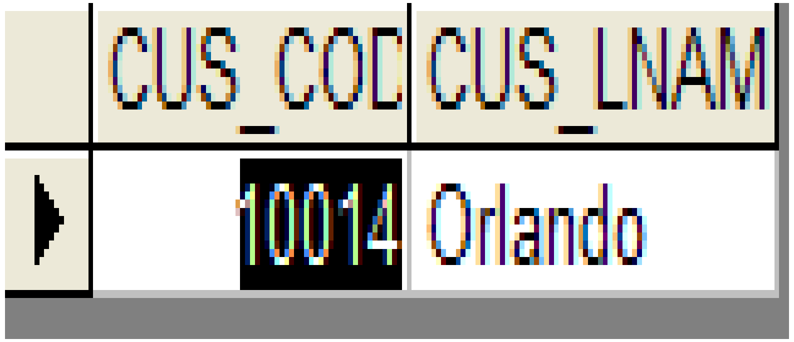
FIGURE 8.16 Multirow subquery operator example



```
SQL> SELECT P_CODE, P_QOH*P_PRICE
  2  FROM PRODUCT
  3  WHERE P_QOH*P_PRICE > ALL
  4  (SELECT P_QOH*P_PRICE FROM PRODUCT
  5   WHERE V_CODE IN (SELECT V_CODE FROM VENDOR WHERE V_STATE = 'FL'));

P_CODE     P_QOH*P_PRICE
---------- -------------
89-WRE-Q         2826.89
```

Database Principles: Design, Implementation, & Management,
10th Edition, Coronel, Morris and Rob

FROM Subqueries (not examinable)



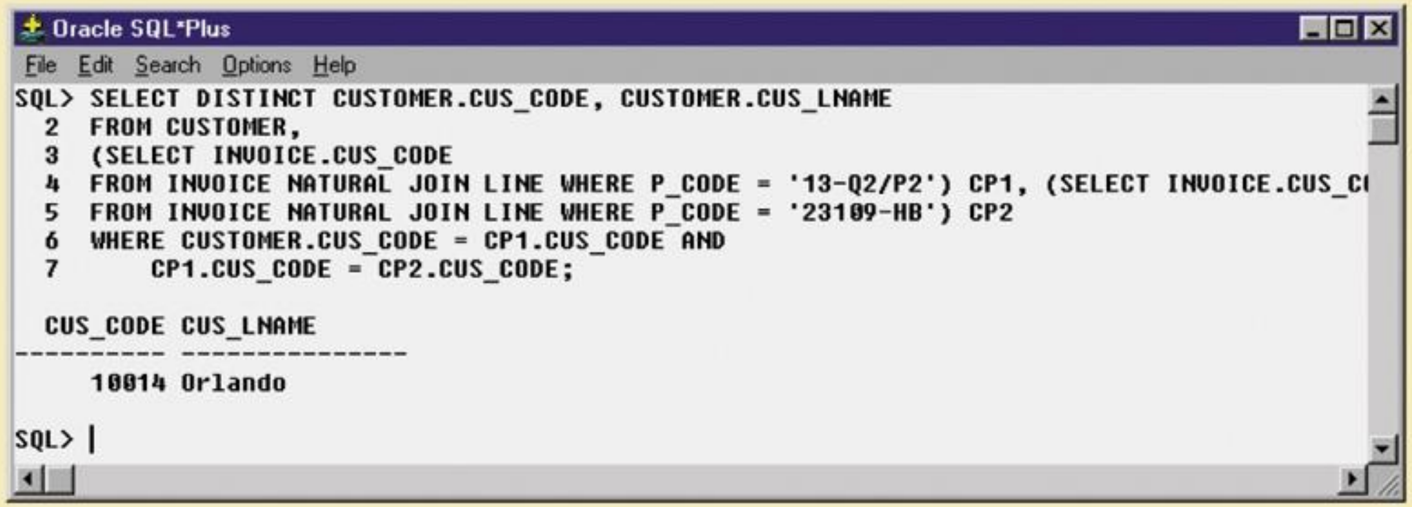
	CUS_COD	CUS_LNAM
▶	10014	Orlando

```
SELECT DISTINCT CUSTOMER.CUS_CODE, CUSTOMER.CUS_LNAME
FROM CUSTOMER, [SELECT INVOICE.CUS_CODE FROM INVOICE, LINE WHERE
INVOICE.INV_NUMBER = LINE.INV_NUMBER AND P_CODE = '13-Q2/P2']. AS CP1,
[SELECT INVOICE.CUS_CODE FROM INVOICE, LINE  WHERE INVOICE.INV_NUMBER =
LINE.INV_NUMBER AND  P_CODE = '23109-HB']. AS CP2
WHERE CUSTOMER.CUS_CODE = CP1.CUS_CODE AND CP1.CUS_CODE =
CP2.CUS_CODE;
```

Database Principles: Design,
Implementation, & Management, 10th
Edition, Coronel, Morris and Rob

FROM Subqueries (not examinable)

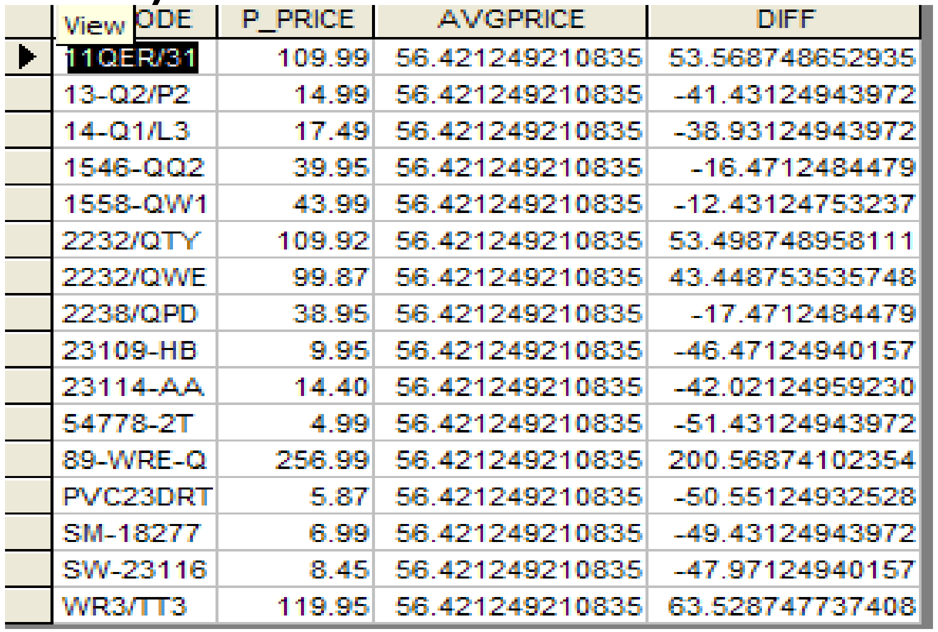
FIGURE 8.17 FROM subquery example



```
SQL> SELECT DISTINCT CUSTOMER.CUS_CODE, CUSTOMER.CUS_LNAME
     FROM CUSTOMER,
     (SELECT INVOICE.CUS_CODE
     FROM INVOICE NATURAL JOIN LINE WHERE P_CODE = '13-Q2/P2') CP1, (SELECT INVOICE.CUS_C
     FROM INVOICE NATURAL JOIN LINE WHERE P_CODE = '23109-HB') CP2
     WHERE CUSTOMER.CUS_CODE = CP1.CUS_CODE AND
         CP1.CUS_CODE = CP2.CUS_CODE;

  CUS_CODE CUS_LNAME
---------- ---------------
     10014 Orlando

SQL> |
```

Database Principles: Design, Implementation, & Management,
10th Edition, Coronel, Morris and Rob

Attribute List Subqueries (not examinable)



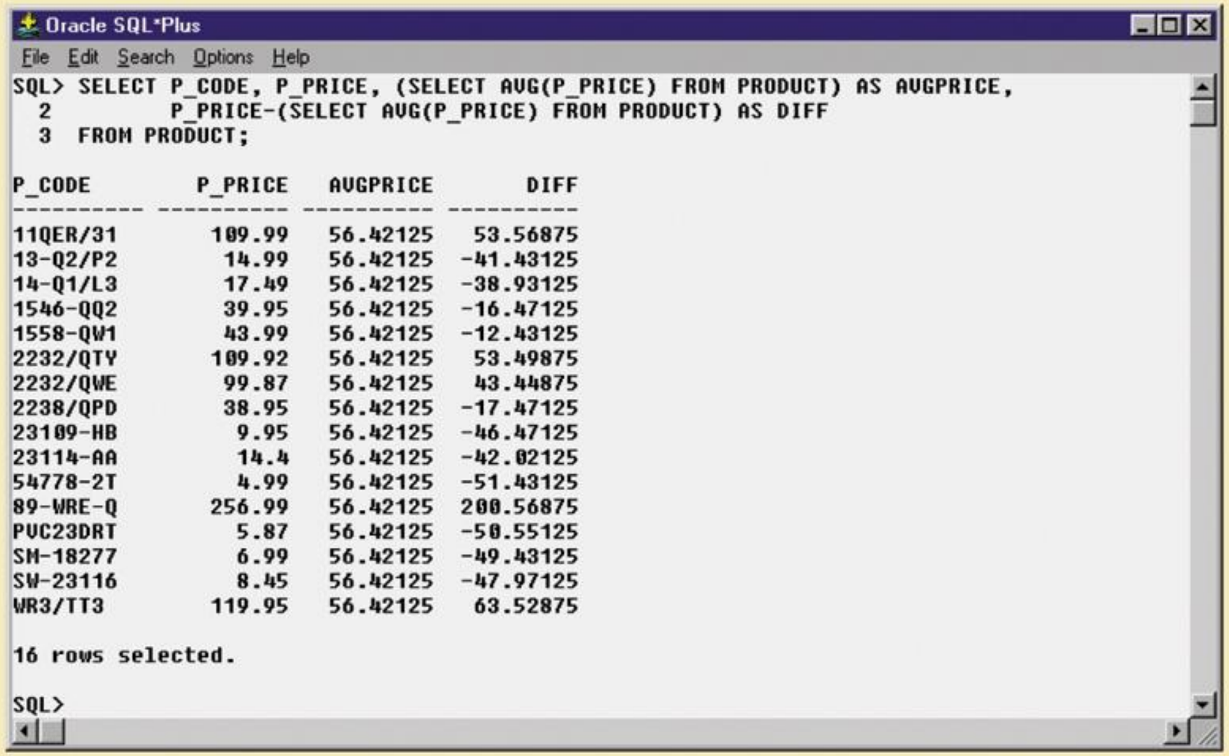
ODE	P_PRICE	AVGPRICE	DIFF
11QER/31	109.99	56.421249210835	53.568748652935
13-Q2/P2	14.99	56.421249210835	-41.43124943972
14-Q1/L3	17.49	56.421249210835	-38.93124943972
1546-QQ2	39.95	56.421249210835	-16.4712484479
1558-QW1	43.99	56.421249210835	-12.43124753237
2232/QTY	109.92	56.421249210835	53.498748958111
2232/QWE	99.87	56.421249210835	43.448753535748
2238/QPD	38.95	56.421249210835	-17.4712484479
23109-HB	9.95	56.421249210835	-46.47124940157
23114-AA	14.40	56.421249210835	-42.02124959230
54778-2T	4.99	56.421249210835	-51.43124943972
89-WRE-Q	256.99	56.421249210835	200.56874102354
PVC23DRT	5.87	56.421249210835	-50.55124932528
SM-18277	6.99	56.421249210835	-49.43124943972
SW-23116	8.45	56.421249210835	-47.97124940157
WR3/TT3	119.95	56.421249210835	63.528747737408

SELECT PRODUCT.P_CODE, PRODUCT.P_PRICE, (SELECT AVG(P_PRICE) FROM PRODUCT) AS AVGPRICE, P_PRICE-(SELECT AVG(P_PRICE) FROM PRODUCT) AS DIFF
FROM PRODUCT;

Database Principles: Design, Implementation, & Management, 10th Edition, Coronel, Morris and Rob

Attribute List Subqueries

FIGURE 8.18 Inline subquery example

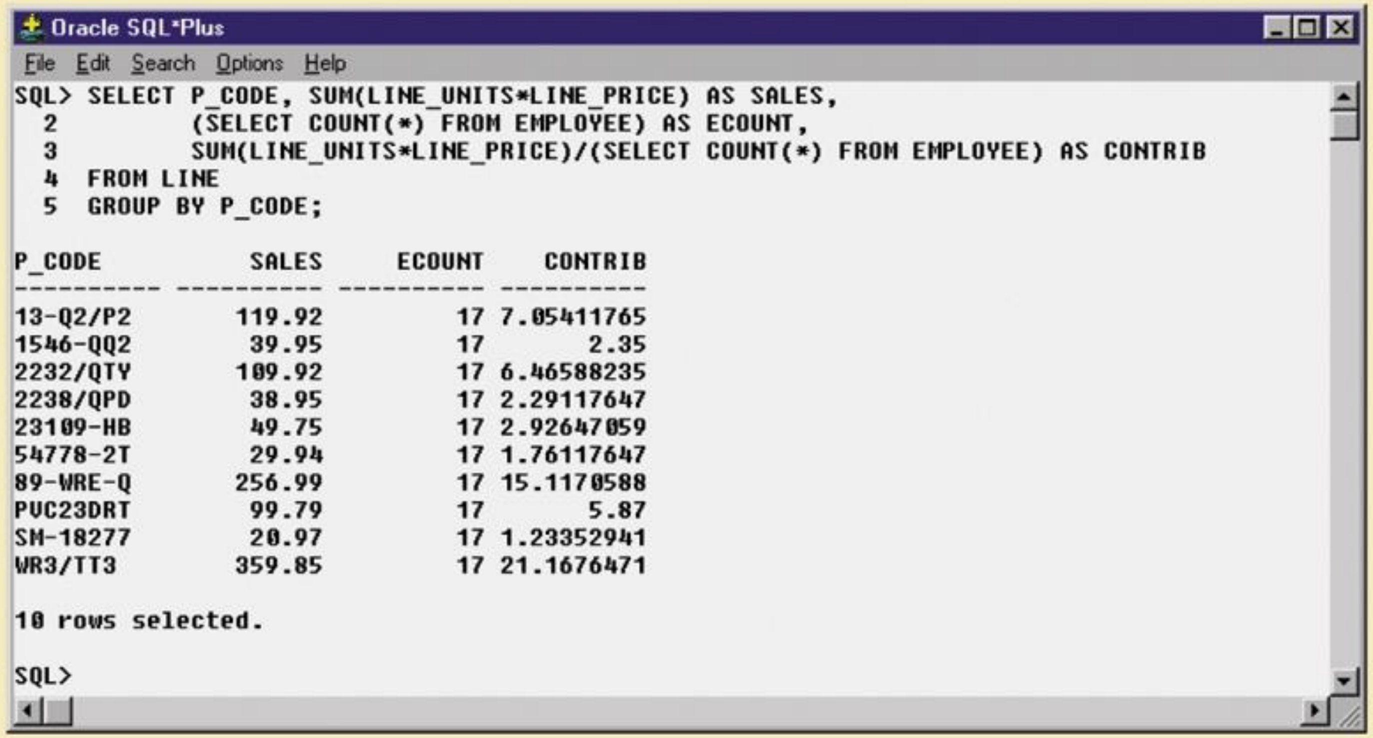


```
SQL> SELECT P_CODE, P_PRICE, (SELECT AVG(P_PRICE) FROM PRODUCT) AS AVGPRICE,
  2         P_PRICE-(SELECT AVG(P_PRICE) FROM PRODUCT) AS DIFF
  3  FROM PRODUCT;

P_CODE        P_PRICE   AVGPRICE       DIFF
---------- ---------- ---------- ----------
11QER/31       109.99   56.42125   53.56875
13-Q2/P2        14.99   56.42125  -41.43125
14-Q1/L3        17.49   56.42125  -38.93125
1546-QQ2        39.95   56.42125  -16.47125
1558-QW1        43.99   56.42125  -12.43125
2232/QTY       109.92   56.42125   53.49875
2232/QWE        99.87   56.42125   43.44875
2238/QPD        38.95   56.42125  -17.47125
23109-HB         9.95   56.42125  -46.47125
23114-AA         14.4   56.42125  -42.02125
54778-2T         4.99   56.42125  -51.43125
89-WRE-Q       256.99   56.42125  200.56875
PVC23DRT         5.87   56.42125  -50.55125
SM-18277         6.99   56.42125  -49.43125
SW-23116         8.45   56.42125  -47.97125
WR3/TT3        119.95   56.42125   63.52875

16 rows selected.

SQL>
```

Database Principles: Design, Implementation, & Management,
10th Edition, Coronel, Morris and Rob

Attribute List Subqueries (continued)

FIGURE 8.19 **Another example of an inline subquery**



```
SQL> SELECT P_CODE, SUM(LINE_UNITS*LINE_PRICE) AS SALES,
  2         (SELECT COUNT(*) FROM EMPLOYEE) AS ECOUNT,
  3         SUM(LINE_UNITS*LINE_PRICE)/(SELECT COUNT(*) FROM EMPLOYEE) AS CONTRIB
  4  FROM LINE
  5  GROUP BY P_CODE;

P_CODE          SALES     ECOUNT    CONTRIB
---------- ---------- ---------- ----------
13-Q2/P2       119.92         17 7.05411765
1546-QQ2        39.95         17       2.35
2232/QTY       109.92         17 6.46588235
2238/QPD        38.95         17 2.29117647
23109-HB        49.75         17 2.92647059
54778-2T        29.94         17 1.76117647
89-WRE-Q       256.99         17 15.1170588
PVC23DRT        99.79         17       5.87
SM-18277        20.97         17 1.23352941
WR3/TT3        359.85         17 21.1676471

10 rows selected.

SQL>
```

Date and Time Functions (self study- not examinable)

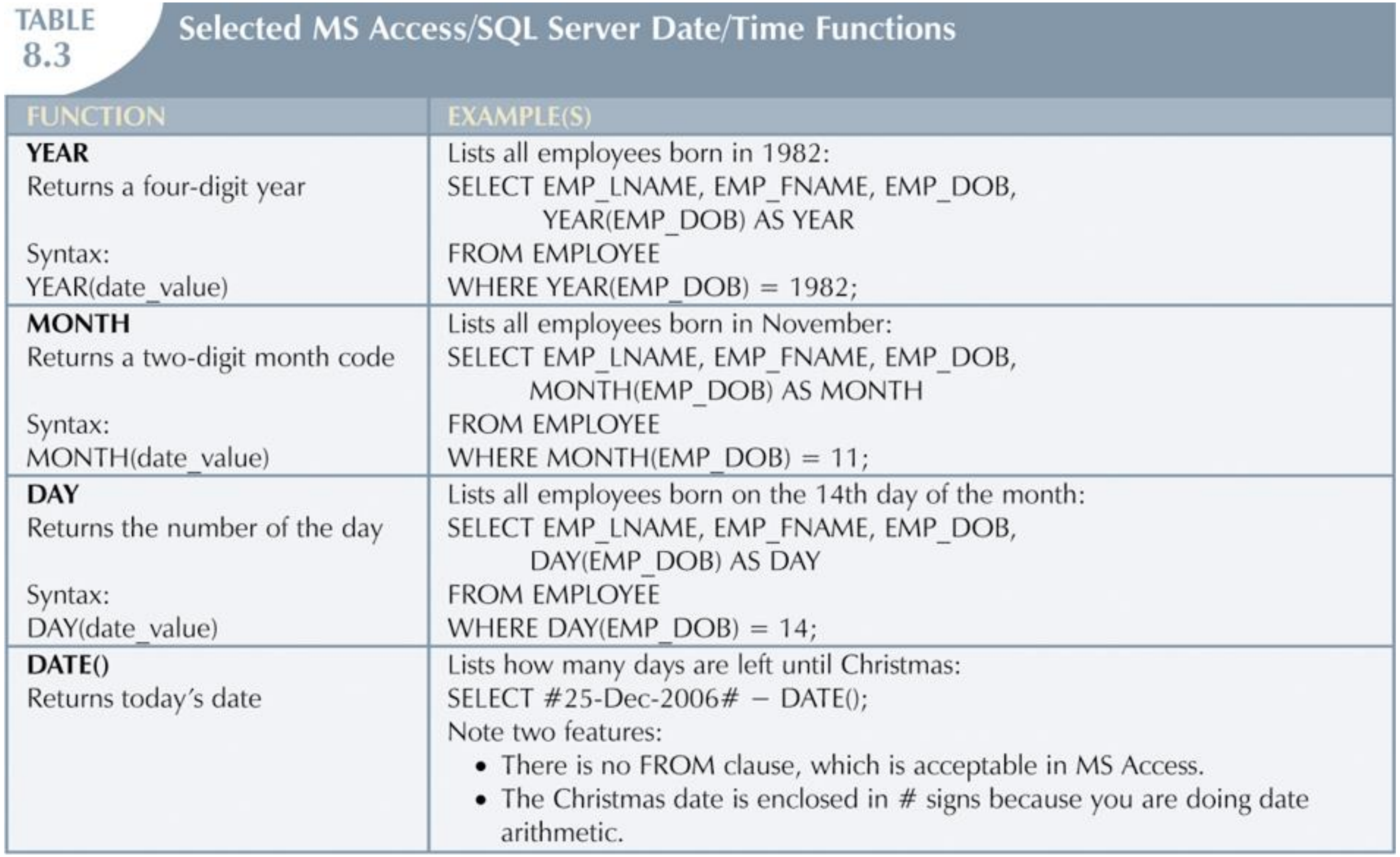


TABLE 8.3 Selected MS Access/SQL Server Date/Time Functions

FUNCTION	EXAMPLE(S)

YEAR
Returns a four-digit year
Syntax:
YEAR(date_value)
	Lists all employees born in 1982:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	YEAR(EMP_DOB) AS YEAR
	FROM EMPLOYEE
	WHERE YEAR(EMP_DOB) = 1982;

MONTH
Returns a two-digit month code
Syntax:
MONTH(date_value)
	Lists all employees born in November:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	MONTH(EMP_DOB) AS MONTH
	FROM EMPLOYEE
	WHERE MONTH(EMP_DOB) = 11;

DAY
Returns the number of the day
Syntax:
DAY(date_value)
	Lists all employees born on the 14th day of the month:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	DAY(EMP_DOB) AS DAY
	FROM EMPLOYEE
	WHERE DAY(EMP_DOB) = 14;

DATE() Returns today's date	Lists how many days are left until Christmas: SELECT #25-Dec-2006# − DATE(); Note two features: • There is no FROM clause, which is acceptable in MS Access. • The Christmas date is enclosed in # signs because you are doing date arithmetic.

Date and Time Functions (continued)

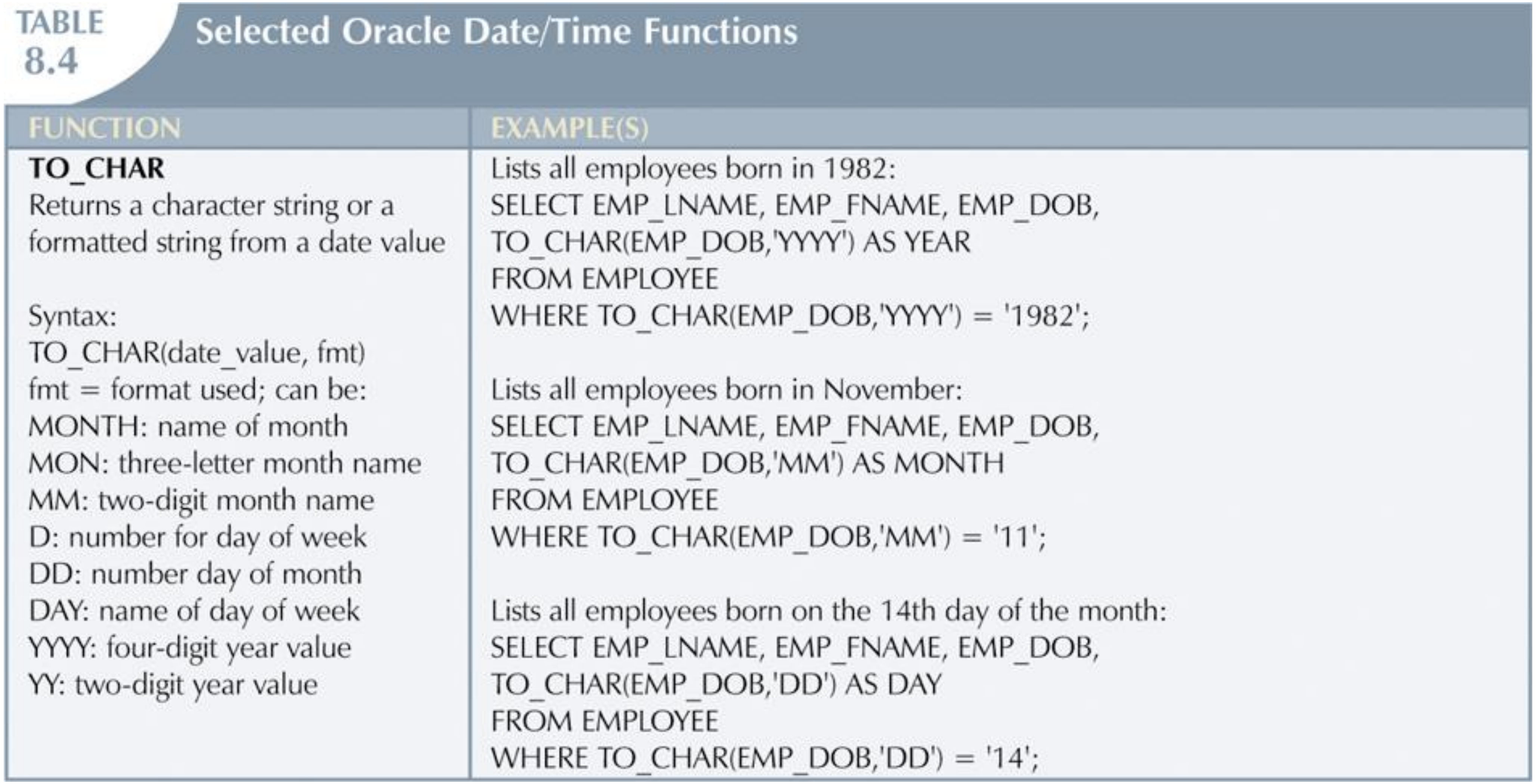


TABLE 8.4 Selected Oracle Date/Time Functions

FUNCTION	EXAMPLE(S)

TO_CHAR
Returns a character string or a formatted string from a date value
 	
Syntax:
TO_CHAR(date_value, fmt)
fmt = format used; can be:
MONTH: name of month
MON: three-letter month name
MM: two-digit month name
D: number for day of week
DD: number day of month
DAY: name of day of week
YYYY: four-digit year value
YY: two-digit year value
	Lists all employees born in 1982:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	TO_CHAR(EMP_DOB,'YYYY') AS YEAR
	FROM EMPLOYEE
	WHERE TO_CHAR(EMP_DOB,'YYYY') = '1982';
	
	Lists all employees born in November:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	TO_CHAR(EMP_DOB,'MM') AS MONTH
	FROM EMPLOYEE
	WHERE TO_CHAR(EMP_DOB,'MM') = '11';
	
	Lists all employees born on the 14th day of the month:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	TO_CHAR(EMP_DOB,'DD') AS DAY
	FROM EMPLOYEE
	WHERE TO_CHAR(EMP_DOB,'DD') = '14';

Date and Time Functions (continued)

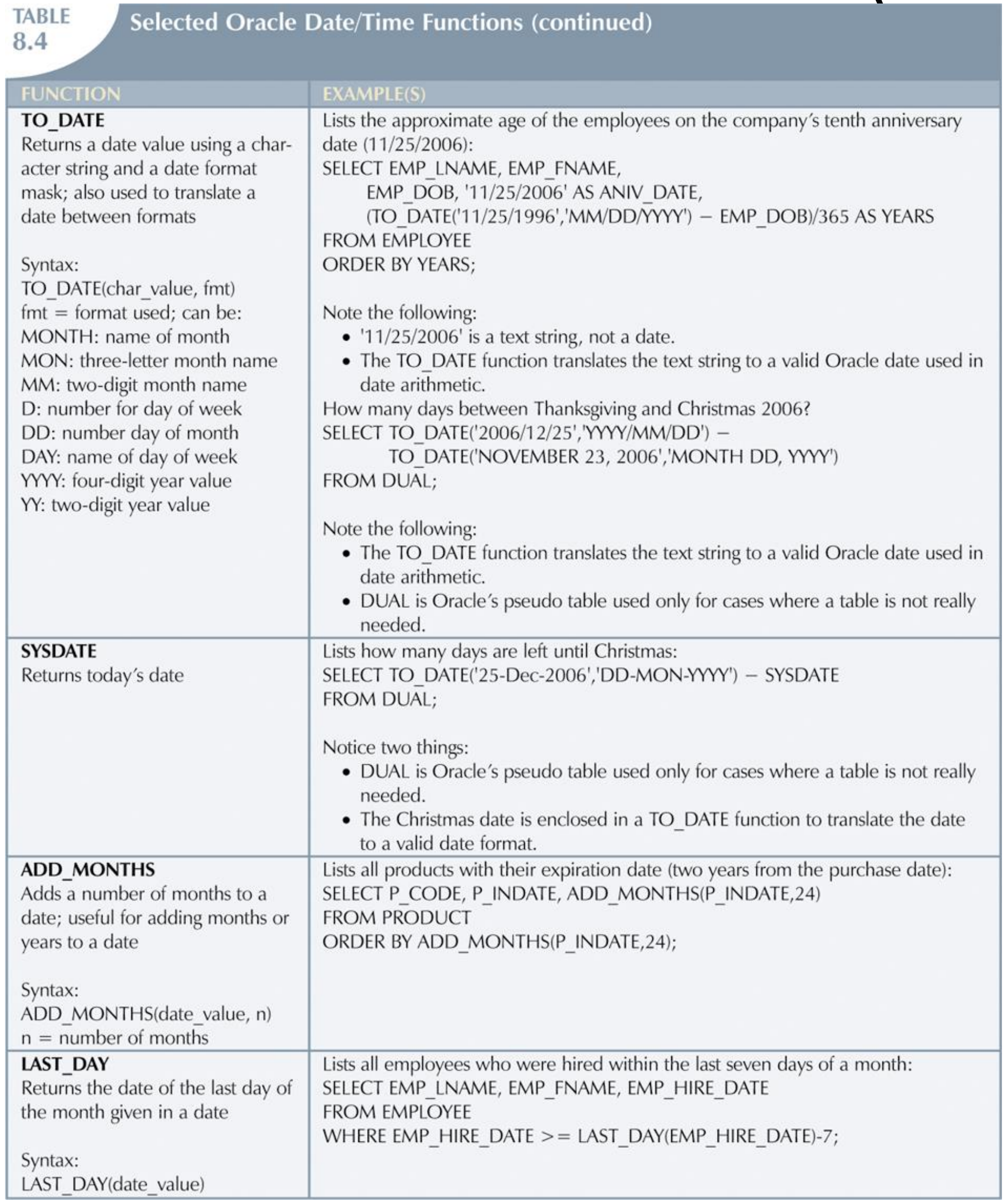


TABLE 8.4 Selected Oracle Date/Time Functions (continued)

FUNCTION	EXAMPLE(S)
TO_DATE Returns a date value using a character string and a date format mask; also used to translate a date between formats Syntax: TO_DATE(char_value, fmt) fmt = format used; can be: MONTH: name of month MON: three-letter month name MM: two-digit month name D: number for day of week DD: number day of month DAY: name of day of week YYYY: four-digit year value YY: two-digit year value	Lists the approximate age of the employees on the company's tenth anniversary date (11/25/2006): SELECT EMP_LNAME, EMP_FNAME, EMP_DOB, '11/25/2006' AS ANIV_DATE, (TO_DATE('11/25/1996','MM/DD/YYYY') – EMP_DOB)/365 AS YEARS FROM EMPLOYEE ORDER BY YEARS; Note the following: • '11/25/2006' is a text string, not a date. • The TO_DATE function translates the text string to a valid Oracle date used in date arithmetic. How many days between Thanksgiving and Christmas 2006? SELECT TO_DATE('2006/12/25','YYYY/MM/DD') – TO_DATE('NOVEMBER 23, 2006','MONTH DD, YYYY') FROM DUAL; Note the following: • The TO_DATE function translates the text string to a valid Oracle date used in date arithmetic. • DUAL is Oracle's pseudo table used only for cases where a table is not really needed.
SYSDATE Returns today's date	Lists how many days are left until Christmas: SELECT TO_DATE('25-Dec-2006','DD-MON-YYYY') – SYSDATE FROM DUAL; Notice two things: • DUAL is Oracle's pseudo table used only for cases where a table is not really needed. • The Christmas date is enclosed in a TO_DATE function to translate the date to a valid date format.

ADD_MONTHS
Adds a number of months to a date; useful for adding months or years to a date
 	
Syntax:
ADD_MONTHS(date_value, n)
n = number of months
	Lists all products with their expiration date (two years from the purchase date):
	SELECT P_CODE, P_INDATE, ADD_MONTHS(P_INDATE,24)
	FROM PRODUCT
	ORDER BY ADD_MONTHS(P_INDATE,24);

LAST_DAY
Returns the date of the last day of the month given in a date
 	
Syntax:
LAST_DAY(date_value)
	Lists all employees who were hired within the last seven days of a month:
	SELECT EMP_LNAME, EMP_FNAME, EMP_HIRE_DATE
	FROM EMPLOYEE
	WHERE EMP_HIRE_DATE >= LAST_DAY(EMP_HIRE_DATE)-7;

Numeric Functions

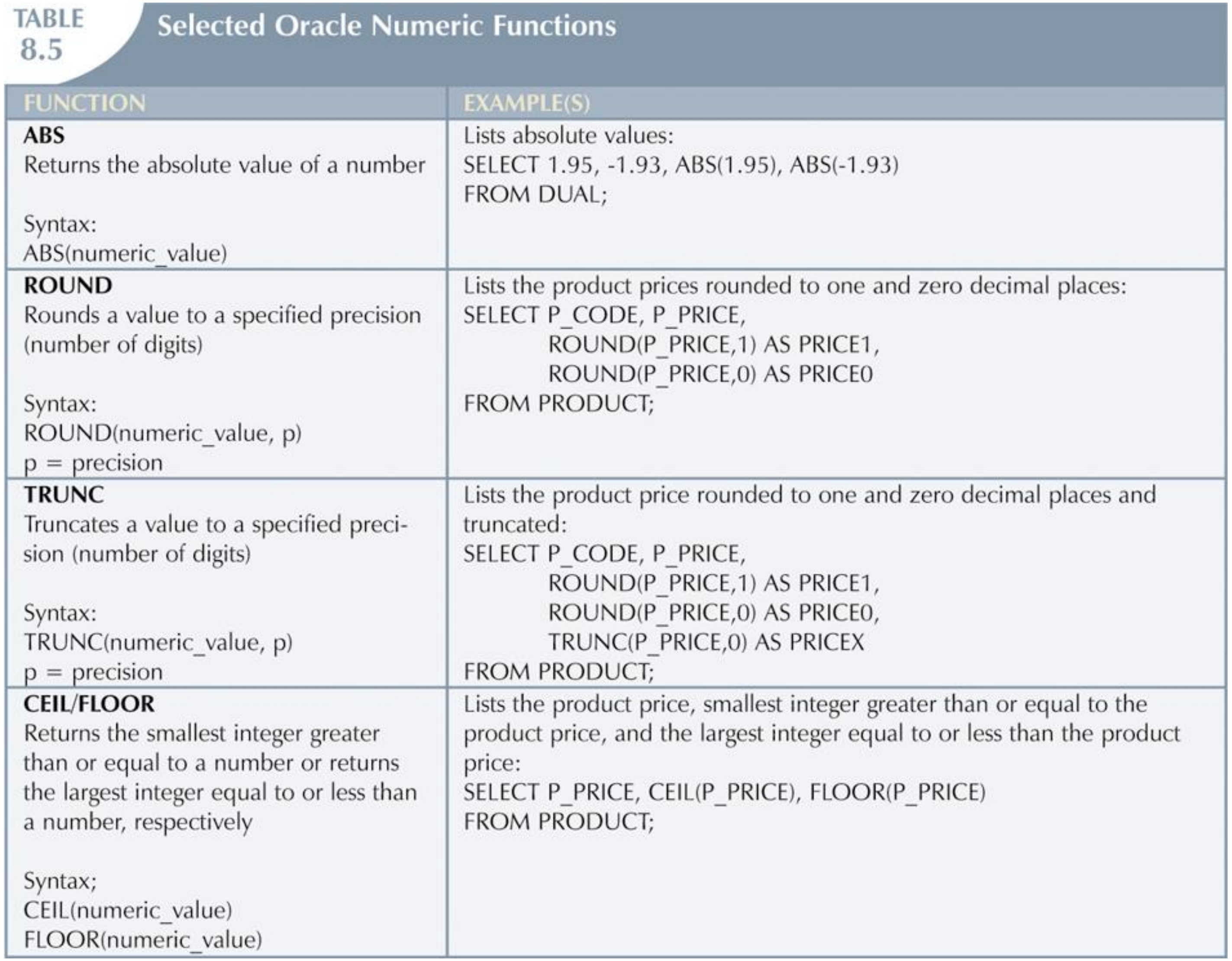


TABLE 8.5 Selected Oracle Numeric Functions

FUNCTION	EXAMPLE(S)

ABS
Returns the absolute value of a number
 	
Syntax:
ABS(numeric_value)
	Lists absolute values:
	SELECT 1.95, -1.93, ABS(1.95), ABS(-1.93)
	FROM DUAL;

ROUND Rounds a value to a specified precision (number of digits) Syntax: ROUND(numeric_value, p) p = precision	Lists the product prices rounded to one and zero decimal places: SELECT P_CODE, P_PRICE, ROUND(P_PRICE,1) AS PRICE1, ROUND(P_PRICE,0) AS PRICE0 FROM PRODUCT;
TRUNC Truncates a value to a specified precision (number of digits) Syntax: TRUNC(numeric_value, p) p = precision	Lists the product price rounded to one and zero decimal places and truncated: SELECT P_CODE, P_PRICE, ROUND(P_PRICE,1) AS PRICE1, ROUND(P_PRICE,0) AS PRICE0, TRUNC(P_PRICE,0) AS PRICEX FROM PRODUCT;

CEIL/FLOOR
Returns the smallest integer greater than or equal to a number or returns the largest integer equal to or less than a number, respectively
 	
Syntax;
CEIL(numeric_value)
FLOOR(numeric_value)
	Lists the product price, smallest integer greater than or equal to the product price, and the largest integer equal to or less than the product price:
	SELECT P_PRICE, CEIL(P_PRICE), FLOOR(P_PRICE)
	FROM PRODUCT;

String Functions

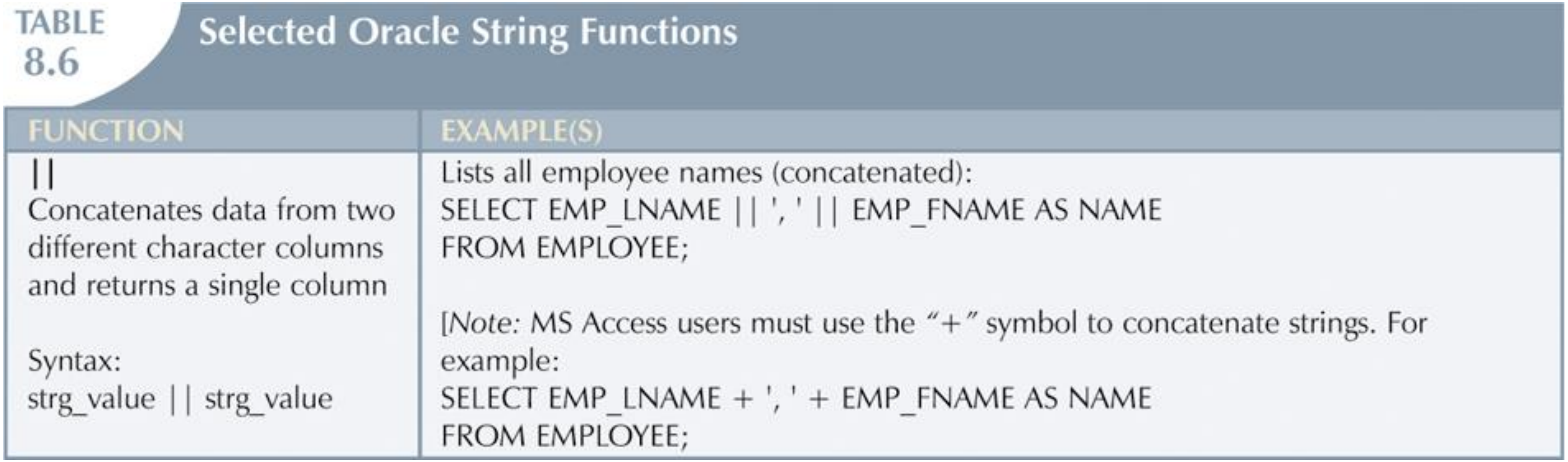


TABLE 8.6 **Selected Oracle String Functions**

FUNCTION	EXAMPLE(S)

||
Concatenates data from two different character columns and returns a single column
Syntax:
strg_value || strg_value
	Lists all employee names (concatenated):
	SELECT EMP_LNAME || ', ' || EMP_FNAME AS NAME
	FROM EMPLOYEE;
	[*Note:* MS Access users must use the "+" symbol to concatenate strings. For example:
	SELECT EMP_LNAME + ', ' + EMP_FNAME AS NAME
	FROM EMPLOYEE;

Database Principles: Design, Implementation, & Management, 10th Edition, Coronel, Morris and Rob

String Functions (continued)

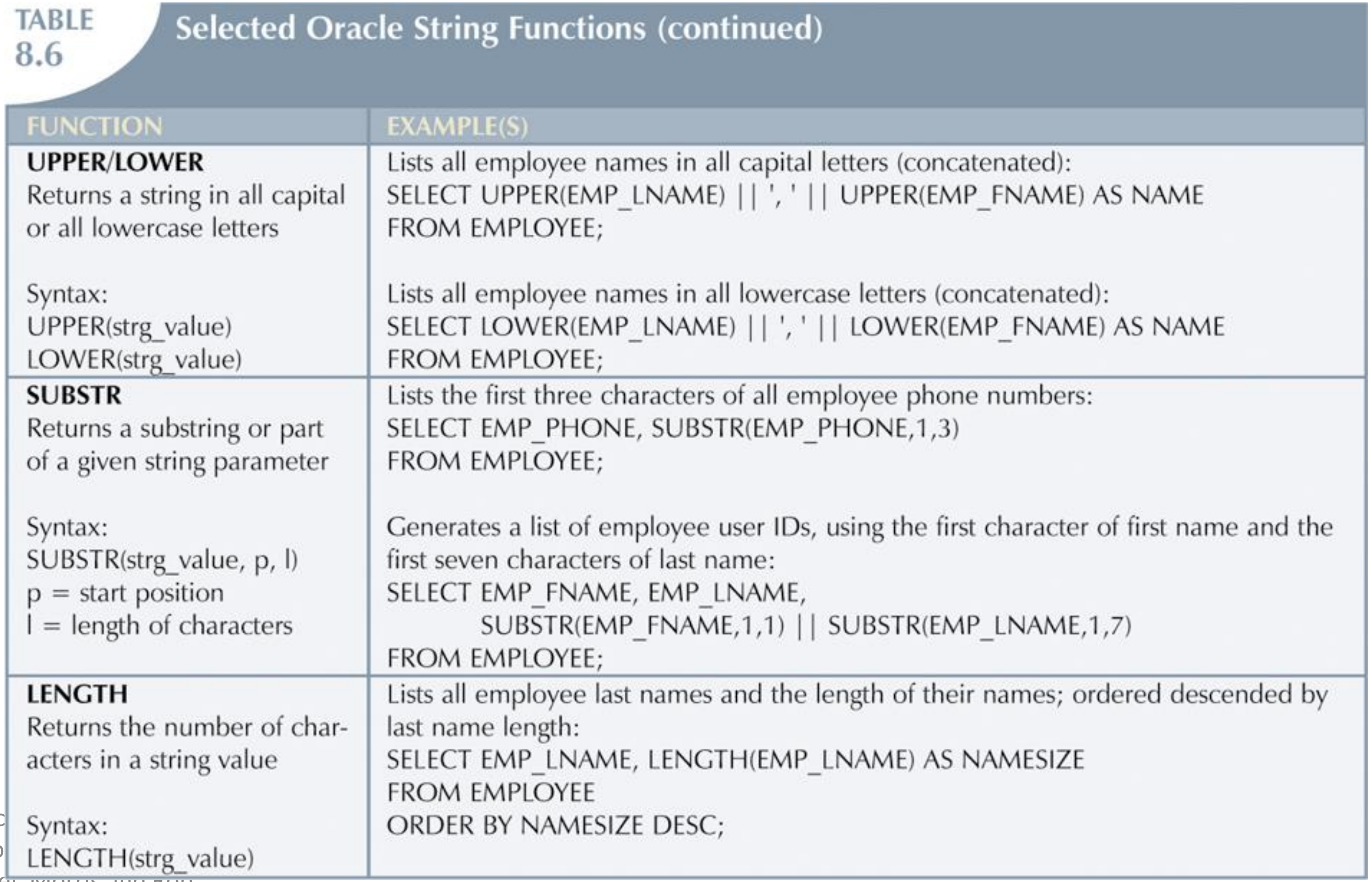


TABLE 8.6 Selected Oracle String Functions (continued)

FUNCTION	EXAMPLE(S)

UPPER/LOWER
Returns a string in all capital or all lowercase letters
 	
Syntax:
UPPER(strg_value)
LOWER(strg_value)
	Lists all employee names in all capital letters (concatenated):
	SELECT UPPER(EMP_LNAME) || ', ' || UPPER(EMP_FNAME) AS NAME
	FROM EMPLOYEE;
	
	Lists all employee names in all lowercase letters (concatenated):
	SELECT LOWER(EMP_LNAME) || ', ' || LOWER(EMP_FNAME) AS NAME
	FROM EMPLOYEE;

SUBSTR
Returns a substring or part of a given string parameter
 	
Syntax:
SUBSTR(strg_value, p, l)
p = start position
l = length of characters
	Lists the first three characters of all employee phone numbers:
	SELECT EMP_PHONE, SUBSTR(EMP_PHONE,1,3)
	FROM EMPLOYEE;
	
	Generates a list of employee user IDs, using the first character of first name and the first seven characters of last name:
	SELECT EMP_FNAME, EMP_LNAME,
	SUBSTR(EMP_FNAME,1,1) || SUBSTR(EMP_LNAME,1,7)
	FROM EMPLOYEE;

LENGTH
Returns the number of characters in a string value
 	
Syntax:
LENGTH(strg_value)
	Lists all employee last names and the length of their names; ordered descended by last name length:
	SELECT EMP_LNAME, LENGTH(EMP_LNAME) AS NAMESIZE
	FROM EMPLOYEE
	ORDER BY NAMESIZE DESC;

Database Princ
Implementatio
Edition, Coronel, Morris and Rob

Conversion Functions

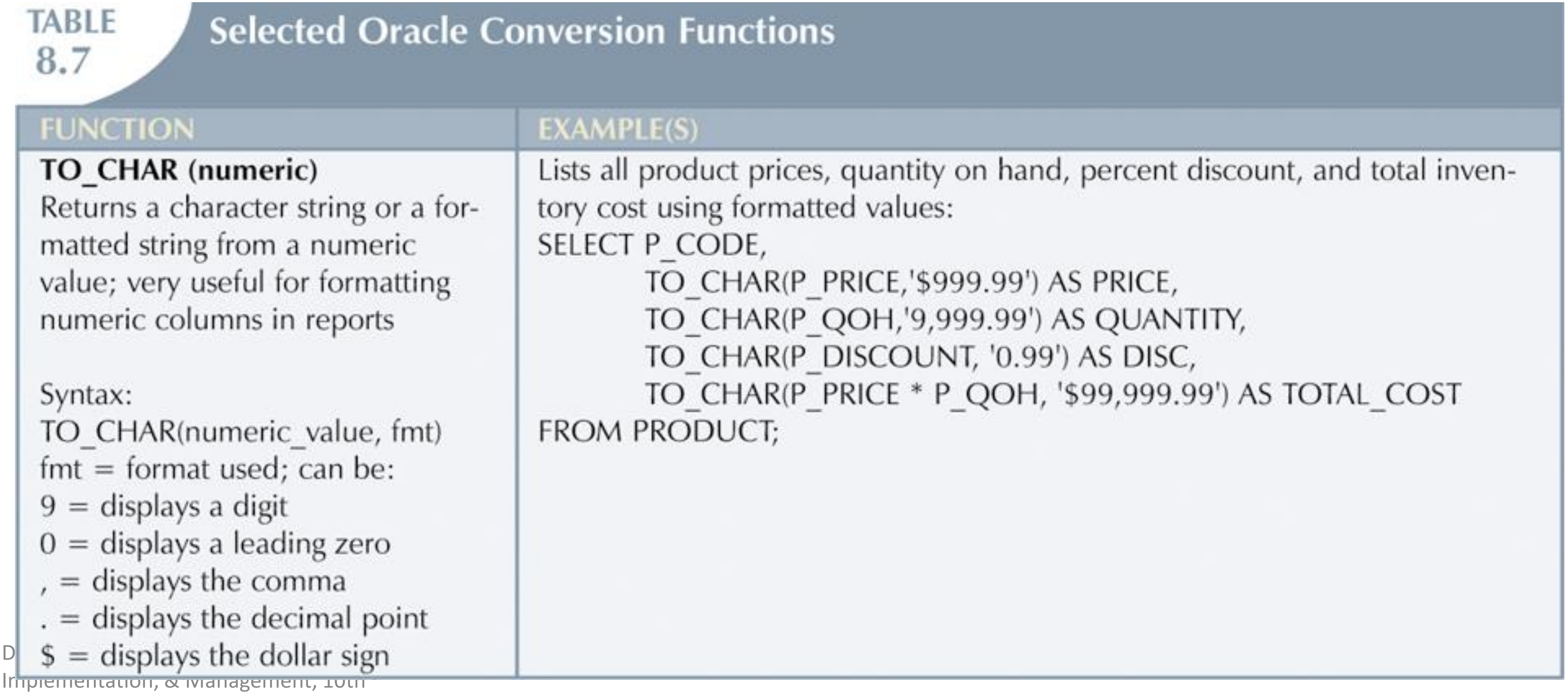


TABLE 8.7 **Selected Oracle Conversion Functions**

FUNCTION	EXAMPLE(S)
TO_CHAR (numeric) Returns a character string or a formatted string from a numeric value; very useful for formatting numeric columns in reports Syntax: TO_CHAR(numeric_value, fmt) fmt = format used; can be: 9 = displays a digit 0 = displays a leading zero , = displays the comma . = displays the decimal point $ = displays the dollar sign	Lists all product prices, quantity on hand, percent discount, and total inventory cost using formatted values: SELECT P_CODE, TO_CHAR(P_PRICE,'$999.99') AS PRICE, TO_CHAR(P_QOH,'9,999.99') AS QUANTITY, TO_CHAR(P_DISCOUNT, '0.99') AS DISC, TO_CHAR(P_PRICE * P_QOH, '$99,999.99') AS TOTAL_COST FROM PRODUCT;

D
Implementation, & Management, 10th
Edition, Coronel, Morris and Rob

Conversion Functions (continued)

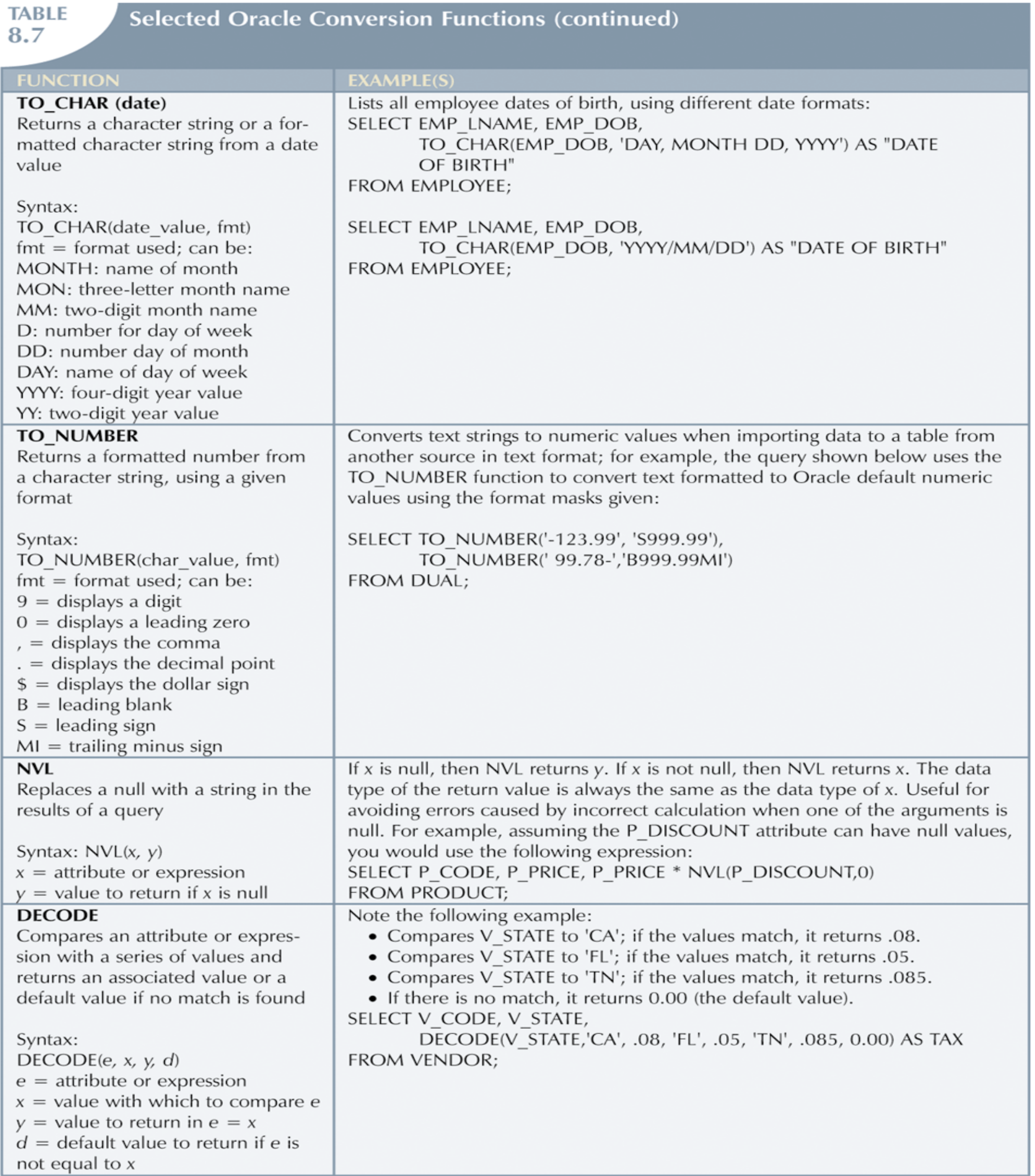


TABLE 8.7 Selected Oracle Conversion Functions (continued)

FUNCTION	EXAMPLE(S)

TO_CHAR (date)
Returns a character string or a formatted character string from a date value
 	
Syntax:
TO_CHAR(date_value, fmt)
fmt = format used; can be:
MONTH: name of month
MON: three-letter month name
MM: two-digit month name
D: number for day of week
DD: number day of month
DAY: name of day of week
YYYY: four-digit year value
YY: two-digit year value
	Lists all employee dates of birth, using different date formats:
	SELECT EMP_LNAME, EMP_DOB,
	TO_CHAR(EMP_DOB, 'DAY, MONTH DD, YYYY') AS "DATE OF BIRTH"
	FROM EMPLOYEE;
	
	SELECT EMP_LNAME, EMP_DOB,
	TO_CHAR(EMP_DOB, 'YYYY/MM/DD') AS "DATE OF BIRTH"
	FROM EMPLOYEE;

TO_NUMBER
Returns a formatted number from a character string, using a given format
 	
Syntax:
TO_NUMBER(char_value, fmt)
fmt = format used; can be:
9 = displays a digit
0 = displays a leading zero
, = displays the comma
. = displays the decimal point
$ = displays the dollar sign
B = leading blank
S = leading sign
MI = trailing minus sign
	Converts text strings to numeric values when importing data to a table from another source in text format; for example, the query shown below uses the TO_NUMBER function to convert text formatted to Oracle default numeric values using the format masks given:
	
	SELECT TO_NUMBER('-123.99', 'S999.99'),
	TO_NUMBER(' 99.78-','B999.99MI')
	FROM DUAL;

NVL
Replaces a null with a string in the results of a query
 	
Syntax: NVL(*x*, *y*)
x = attribute or expression
y = value to return if *x* is null
	If *x* is null, then NVL returns *y*. If *x* is not null, then NVL returns *x*. The data type of the return value is always the same as the data type of *x*. Useful for avoiding errors caused by incorrect calculation when one of the arguments is null. For example, assuming the P_DISCOUNT attribute can have null values, you would use the following expression:
	SELECT P_CODE, P_PRICE, P_PRICE * NVL(P_DISCOUNT,0)
	FROM PRODUCT;

DECODE Compares an attribute or expression with a series of values and returns an associated value or a default value if no match is found Syntax: DECODE(*e*, *x*, *y*, *d*) *e* = attribute or expression *x* = value with which to compare *e* *y* = value to return in *e* = *x* *d* = default value to return if *e* is not equal to *x*	Note the following example: • Compares V_STATE to 'CA'; if the values match, it returns .08. • Compares V_STATE to 'FL'; if the values match, it returns .05. • Compares V_STATE to 'TN'; if the values match, it returns .085. • If there is no match, it returns 0.00 (the default value). SELECT V_CODE, V_STATE, DECODE(V_STATE,'CA', .08, 'FL', .05, 'TN', .085, 0.00) AS TAX FROM VENDOR;

Database Principles: Design,
Implementation, & Manager
Edition, Coronel, Morris and

Triggers (Theory only)

- Automating business procedures and automatically maintaining data integrity and consistency are critical in modern business
- Most critical business procedures is proper inventory management
- Eg. Make sure that current product sales can be supported with sufficient product availability
- Therefore, necessary that a product order be sent to a vendor when the product's inventory drops below its minimum allowable quantity on hand

Triggers

- Can be accomplish these task by writing multiple SQL statements: one to update the product quantity on hand and another to update the product reorder flag.
- Such multistage process is inefficient because a series of SQL statements must be written and executed each time a product is sold.
- Also, SQL environment requires that somebody must remember to perform the SQL tasks

Triggers

- A trigger is procedural SQL code that is automatically invoked by the RDBMS upon the occurrence of a given data manipulation event
- Useful to remember:

- A trigger is invoked before or after a data row is inserted, updated, or deleted
- A trigger is associated with a database table
- Each database table may have one or more triggers
- A trigger is executed as part of the transaction that triggered it

Database Principles: Design, Implementation, & Management, 10th Edition, Coronel, Morris and Rob

Triggers

- Triggers are critical to proper database operation and management

- Triggers can be used to enforce constraints that cannot be enforced at the DBMS design and implementation

e.g. require that every invoice have at least one line item

- Triggers add functionality by automating critical actions and providing appropriate warnings and suggestions for remedial actions

- Triggers can be used to update table values, insert records in tables, and call other stored procedures

Database Principles: Design, Implementation, & Management, 10th Edition, Coronel, Morris and Rob

Triggers

- Triggers play a critical role in making the database truly useful. Create triggers for:

- Auditing purposes (creating audit logs)
- Automatic generation of derived column value
- Enforcement of business or security constraints

notify a manager every time an employee's bank account number changes

- Creation of replica tables for backup purposes

SQL SERVER 2005

```
CREATE TRIGGER TRG_PRODUCT_REORDER
ON PRODUCT
FOR INSERT, UPDATE
AS
BEGIN
  IF UPDATE(P_QOH)
  BEGIN
  UPDATE PRODUCT
           SET P_REORDER = 1
           WHERE P_QOH <= P_MIN;
  END
END;
```

```
CREATE TRIGGER TRG_PRODUCT_REORDER
ON PRODUCT
FOR INSERT, UPDATE
AS
BEGIN
  IF UPDATE(P_QOH)
  BEGIN
  UPDATE PRODUCT
                SET P_REORDER = 1
                WHERE P_QOH <= P_MIN;
  END
  IF UPDATE(P_MIN)
  BEGIN
  UPDATE PRODUCT
                SET P_REORDER = 1
                WHERE P_QOH <= P_MIN;
  END
END;
```

<u>SQL SERVER 2005</u>

Stored Procedures (Theory only)

- A stored procedure is a named collection of procedural and SQL statements

Database Principles: Design,
Implementation, & Management, 10th
Edition, Coronel, Morris and Rob

Stored Procedures

- Advantages
    - Substantially reduce network traffic and increase performance
        - No transmission of individual SQL statements over network
    - Help reduce code duplication by means of code isolation and code sharing
        - Minimize chance of errors and cost of application development and maintenance

Database Principles: Design,
Implementation, & Management, 10th
Edition, Coronel, Morris and Rob