Chapter 7

Advanced SQL

Database Systems:
Design, Implementation, and Management,
Sixth Edition, Rob and Coronel

In this chapter, you will learn:

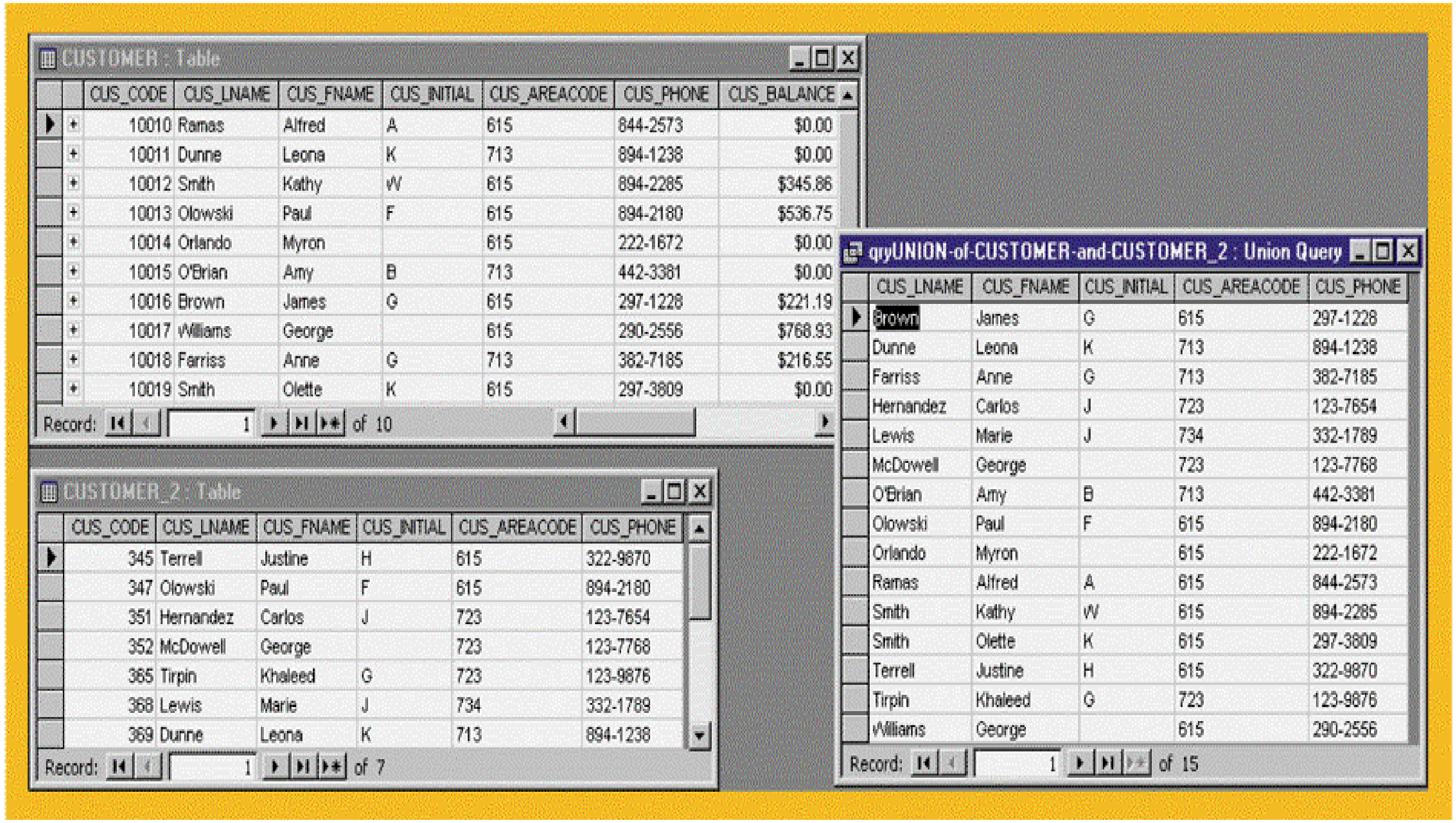
- About the relational set operators UNION, UNION ALL, INTERSECT, and MINUS
- How to use the advanced SQL JOIN operator syntax
- About the different types of subqueries and correlated queries
- How to use SQL functions to manipulate dates, strings, and other data

In this chapter, you will learn: (continued)

- How to create and use updatable views
- How to create and use triggers and stored procedures
- How to create embedded SQL

UNION Query Result

FIGURE 7.1 UNION QUERY RESULT



CUSTOMER : Table

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
10010	Ramas	Alfred	A	615	844-2573	$0.00
10011	Dunne	Leona	K	713	894-1238	$0.00
10012	Smith	Kathy	W	615	894-2285	$345.86
10013	Olowski	Paul	F	615	894-2180	$536.75
10014	Orlando	Myron		615	222-1672	$0.00
10015	O'Brian	Amy	B	713	442-3381	$0.00
10016	Brown	James	G	615	297-1228	$221.19
10017	Williams	George		615	290-2556	$768.93
10018	Farriss	Anne	G	713	382-7185	$216.55
10019	Smith	Olette	K	615	297-3809	$0.00

Record: 1 of 10

CUSTOMER_2 : Table

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
345	Terrell	Justine	H	615	322-9870
347	Olowski	Paul	F	615	894-2180
351	Hernandez	Carlos	J	723	123-7654
352	McDowell	George		723	123-7768
365	Tirpin	Khaleed	G	723	123-9876
368	Lewis	Marie	J	734	332-1789
369	Dunne	Leona	K	713	894-1238

Record: 1 of 7

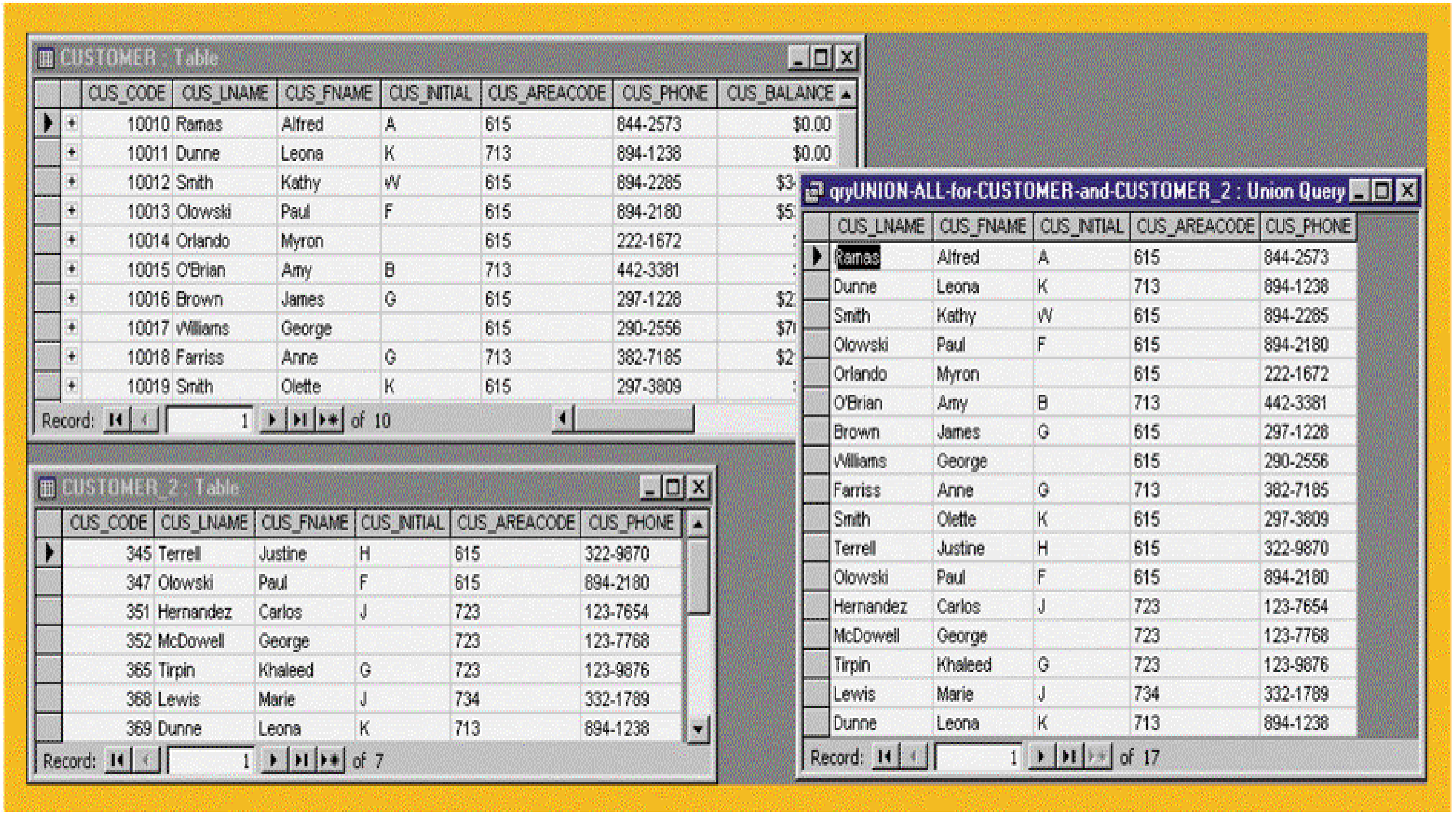
qryUNION-of-CUSTOMER-and-CUSTOMER_2 : Union Query

CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
Brown	James	G	615	297-1228
Dunne	Leona	K	713	894-1238
Farriss	Anne	G	713	382-7185
Hernandez	Carlos	J	723	123-7654
Lewis	Marie	J	734	332-1789
McDowell	George		723	123-7768
O'Brian	Amy	B	713	442-3381
Olowski	Paul	F	615	894-2180
Orlando	Myron		615	222-1672
Ramas	Alfred	A	615	844-2573
Smith	Kathy	W	615	894-2285
Smith	Olette	K	615	297-3809
Terrell	Justine	H	615	322-9870
Tirpin	Khaleed	G	723	123-9876
Williams	George		615	290-2556

Record: 1 of 15

UNION ALL Query Result

FIGURE 7.2 UNION ALL QUERY RESULT



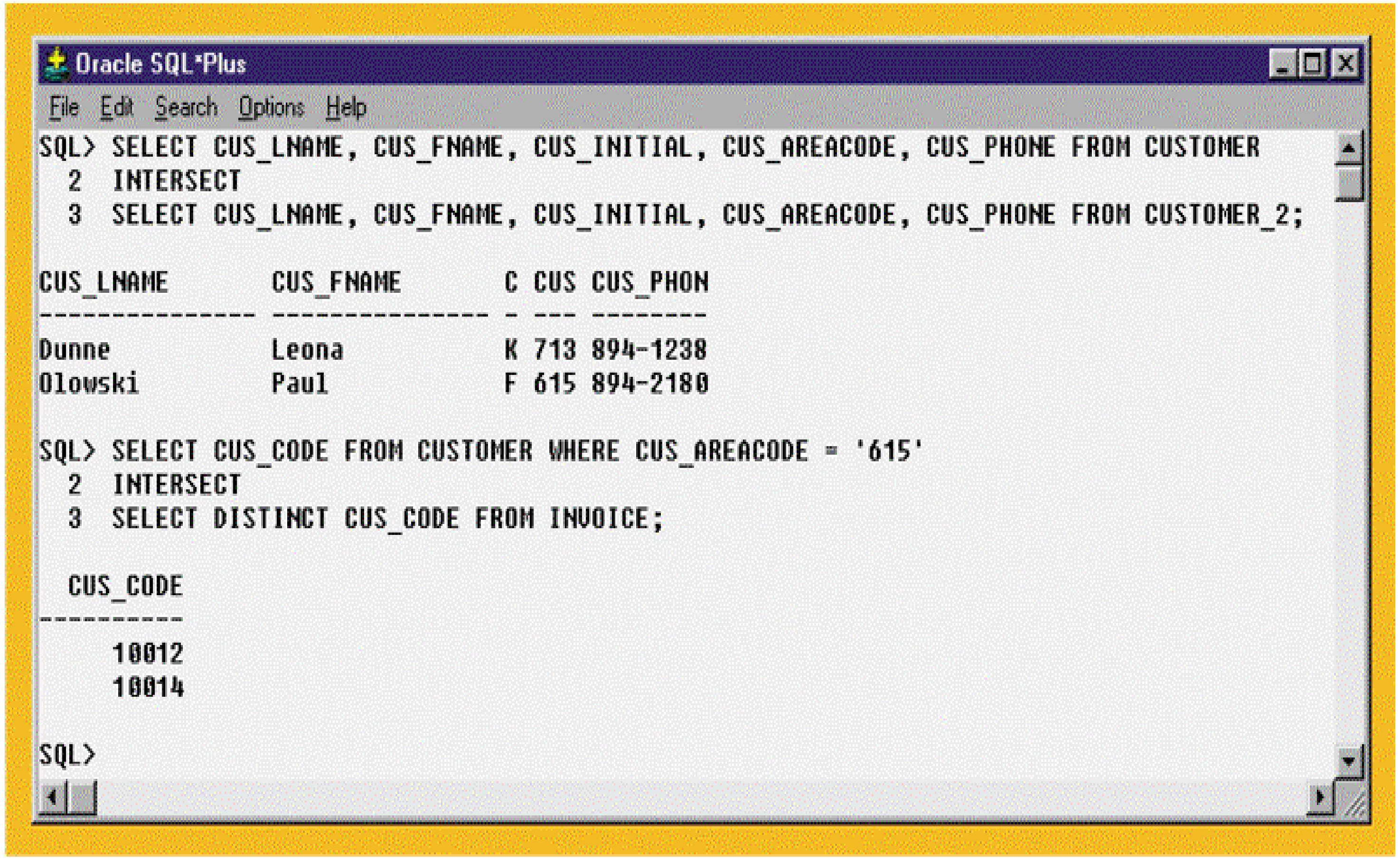
CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
10010	Ramas	Alfred	A	615	844-2573	$0.00
10011	Dunne	Leona	K	713	894-1238	$0.00
10012	Smith	Kathy	W	615	894-2285	$3
10013	Olowski	Paul	F	615	894-2180	$5
10014	Orlando	Myron		615	222-1672	
10015	O'Brian	Amy	B	713	442-3381	
10016	Brown	James	G	615	297-1228	$2
10017	Williams	George		615	290-2556	$7
10018	Farriss	Anne	G	713	382-7185	$2
10019	Smith	Olette	K	615	297-3809	

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
345	Terrell	Justine	H	615	322-9870
347	Olowski	Paul	F	615	894-2180
351	Hernandez	Carlos	J	723	123-7654
352	McDowell	George		723	123-7768
365	Tirpin	Khaleed	G	723	123-9876
368	Lewis	Marie	J	734	332-1789
369	Dunne	Leona	K	713	894-1238

CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
Ramas	Alfred	A	615	844-2573
Dunne	Leona	K	713	894-1238
Smith	Kathy	W	615	894-2285
Olowski	Paul	F	615	894-2180
Orlando	Myron		615	222-1672
O'Brian	Amy	B	713	442-3381
Brown	James	G	615	297-1228
Williams	George		615	290-2556
Farriss	Anne	G	713	382-7185
Smith	Olette	K	615	297-3809
Terrell	Justine	H	615	322-9870
Olowski	Paul	F	615	894-2180
Hernandez	Carlos	J	723	123-7654
McDowell	George		723	123-7768
Tirpin	Khaleed	G	723	123-9876
Lewis	Marie	J	734	332-1789
Dunne	Leona	K	713	894-1238

INTERSECT Query Result

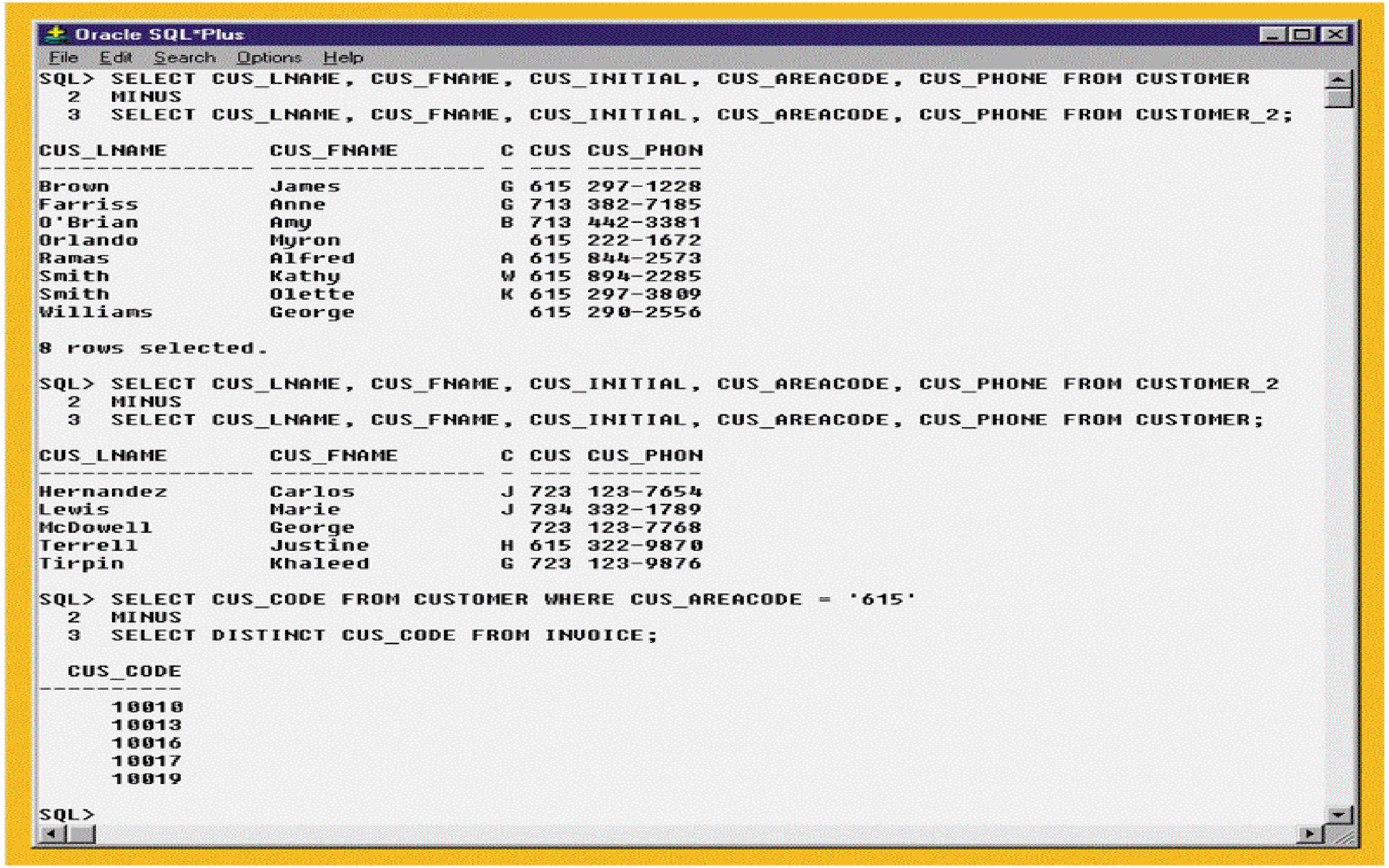
FIGURE 7.3 INTERSECT QUERY RESULT



```
SQL> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE FROM CUSTOMER
  2  INTERSECT
  3  SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE FROM CUSTOMER_2;

CUS_LNAME       CUS_FNAME       C CUS CUS_PHON
--------------- --------------- - --- --------
Dunne           Leona           K 713 894-1238
Olowski         Paul            F 615 894-2180

SQL> SELECT CUS_CODE FROM CUSTOMER WHERE CUS_AREACODE = '615'
  2  INTERSECT
  3  SELECT DISTINCT CUS_CODE FROM INVOICE;

  CUS_CODE
----------
     10012
     10014

SQL>
```

MINUS Query Results

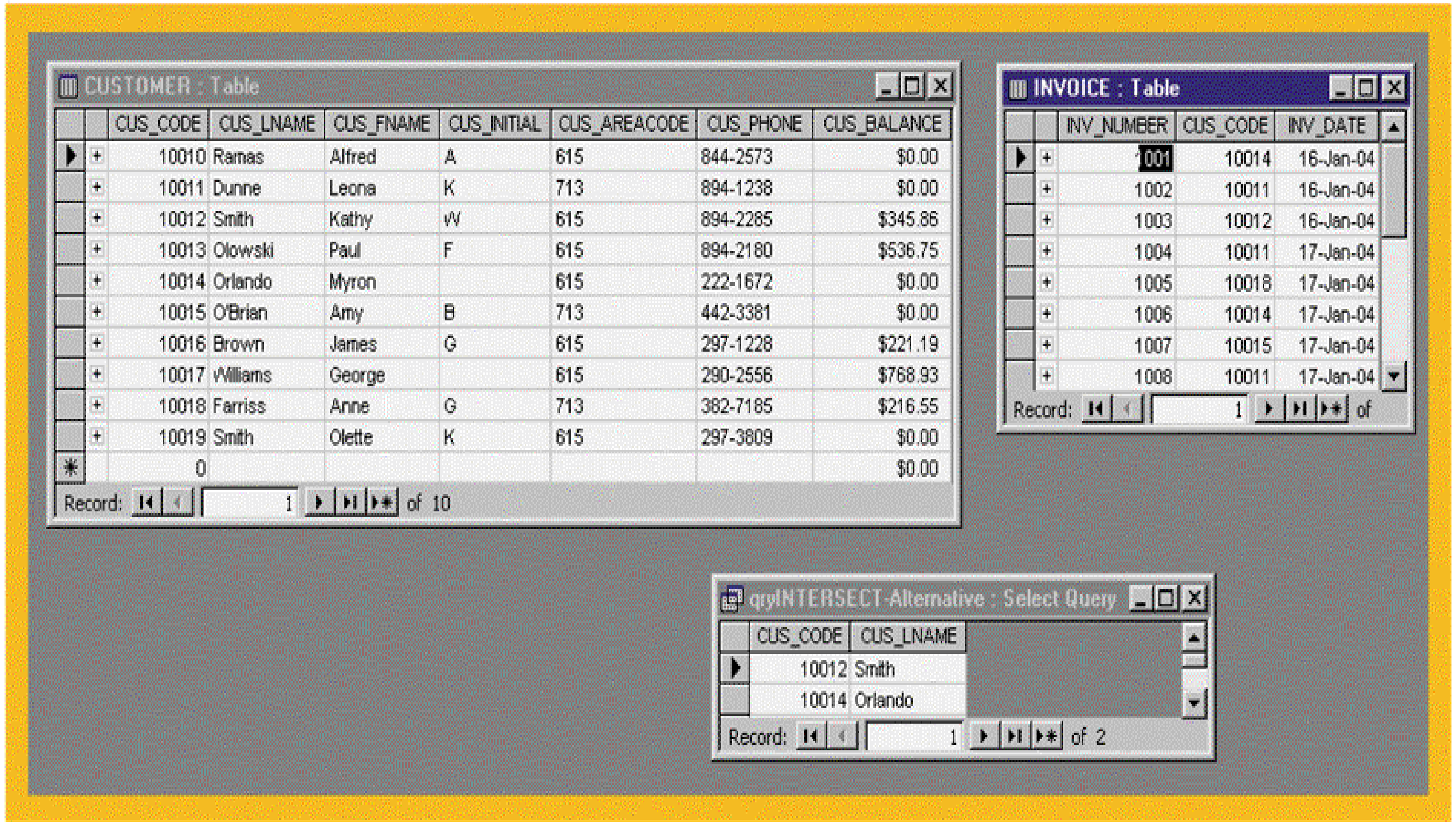
FIGURE 7.4 MINUS QUERY RESULTS



```
SQL> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE FROM CUSTOMER
  2  MINUS
  3  SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE FROM CUSTOMER_2;

CUS_LNAME        CUS_FNAME        C CUS CUS_PHON
---------------- ---------------- - --- --------
Brown            James            G 615 297-1228
Farriss          Anne             G 713 382-7185
O'Brian          Amy              B 713 442-3381
Orlando          Myron              615 222-1672
Ramas            Alfred           A 615 844-2573
Smith            Kathy            W 615 894-2285
Smith            Olette           K 615 297-3809
Williams         George             615 290-2556

8 rows selected.

SQL> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE FROM CUSTOMER_2
  2  MINUS
  3  SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE FROM CUSTOMER;

CUS_LNAME        CUS_FNAME        C CUS CUS_PHON
---------------- ---------------- - --- --------
Hernandez        Carlos           J 723 123-7654
Lewis            Marie            J 734 332-1789
McDowell         George             723 123-7768
Terrell          Justine          H 615 322-9870
Tirpin           Khaleed          G 723 123-9876

SQL> SELECT CUS_CODE FROM CUSTOMER WHERE CUS_AREACODE = '615'
  2  MINUS
  3  SELECT DISTINCT CUS_CODE FROM INVOICE;

  CUS_CODE
----------
     10010
     10013
     10016
     10017
     10019

SQL>
```

INTERSECT Alternative

FIGURE 7.5 INTERSECT ALTERNATIVE



CUSTOMER : Table

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
10010	Ramas	Alfred	A	615	844-2573	$0.00
10011	Dunne	Leona	K	713	894-1238	$0.00
10012	Smith	Kathy	W	615	894-2285	$345.86
10013	Olowski	Paul	F	615	894-2180	$536.75
10014	Orlando	Myron		615	222-1672	$0.00
10015	O'Brian	Amy	B	713	442-3381	$0.00
10016	Brown	James	G	615	297-1228	$221.19
10017	Williams	George		615	290-2556	$768.93
10018	Farriss	Anne	G	713	382-7185	$216.55
10019	Smith	Olette	K	615	297-3809	$0.00
0						$0.00

Record: 1 of 10

INVOICE : Table

INV_NUMBER	CUS_CODE	INV_DATE
1001	10014	16-Jan-04
1002	10011	16-Jan-04
1003	10012	16-Jan-04
1004	10011	17-Jan-04
1005	10018	17-Jan-04
1006	10014	17-Jan-04
1007	10015	17-Jan-04
1008	10011	17-Jan-04

Record: 1 of

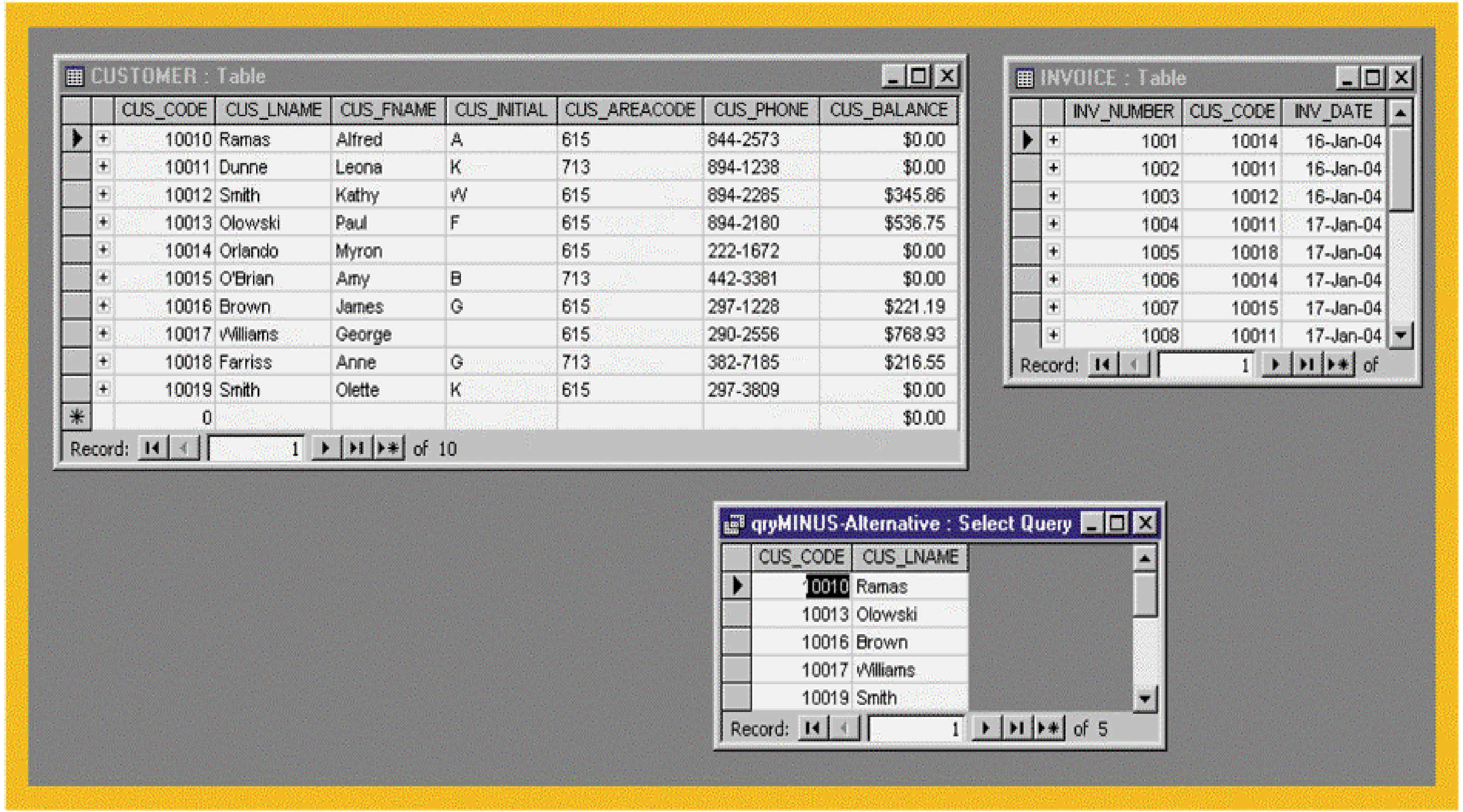
qryINTERSECT-Alternative : Select Query

CUS_CODE	CUS_LNAME
10012	Smith
10014	Orlando

Record: 1 of 2

MINUS Alternative

FIGURE 7.6 MINUS ALTERNATIVE



CUSTOMER : Table

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
10010	Ramas	Alfred	A	615	844-2573	$0.00
10011	Dunne	Leona	K	713	894-1238	$0.00
10012	Smith	Kathy	W	615	894-2285	$345.86
10013	Olowski	Paul	F	615	894-2180	$536.75
10014	Orlando	Myron		615	222-1672	$0.00
10015	O'Brian	Amy	B	713	442-3381	$0.00
10016	Brown	James	G	615	297-1228	$221.19
10017	Williams	George		615	290-2556	$768.93
10018	Farriss	Anne	G	713	382-7185	$216.55
10019	Smith	Olette	K	615	297-3809	$0.00
0						$0.00

Record: 1 of 10

INVOICE : Table

INV_NUMBER	CUS_CODE	INV_DATE
1001	10014	16-Jan-04
1002	10011	16-Jan-04
1003	10012	16-Jan-04
1004	10011	17-Jan-04
1005	10018	17-Jan-04
1006	10014	17-Jan-04
1007	10015	17-Jan-04
1008	10011	17-Jan-04

Record: 1 of

qryMINUS-Alternative : Select Query

CUS_CODE	CUS_LNAME
10010	Ramas
10013	Olowski
10016	Brown
10017	Williams
10019	Smith

Record: 1 of 5

SQL Join Expression Styles

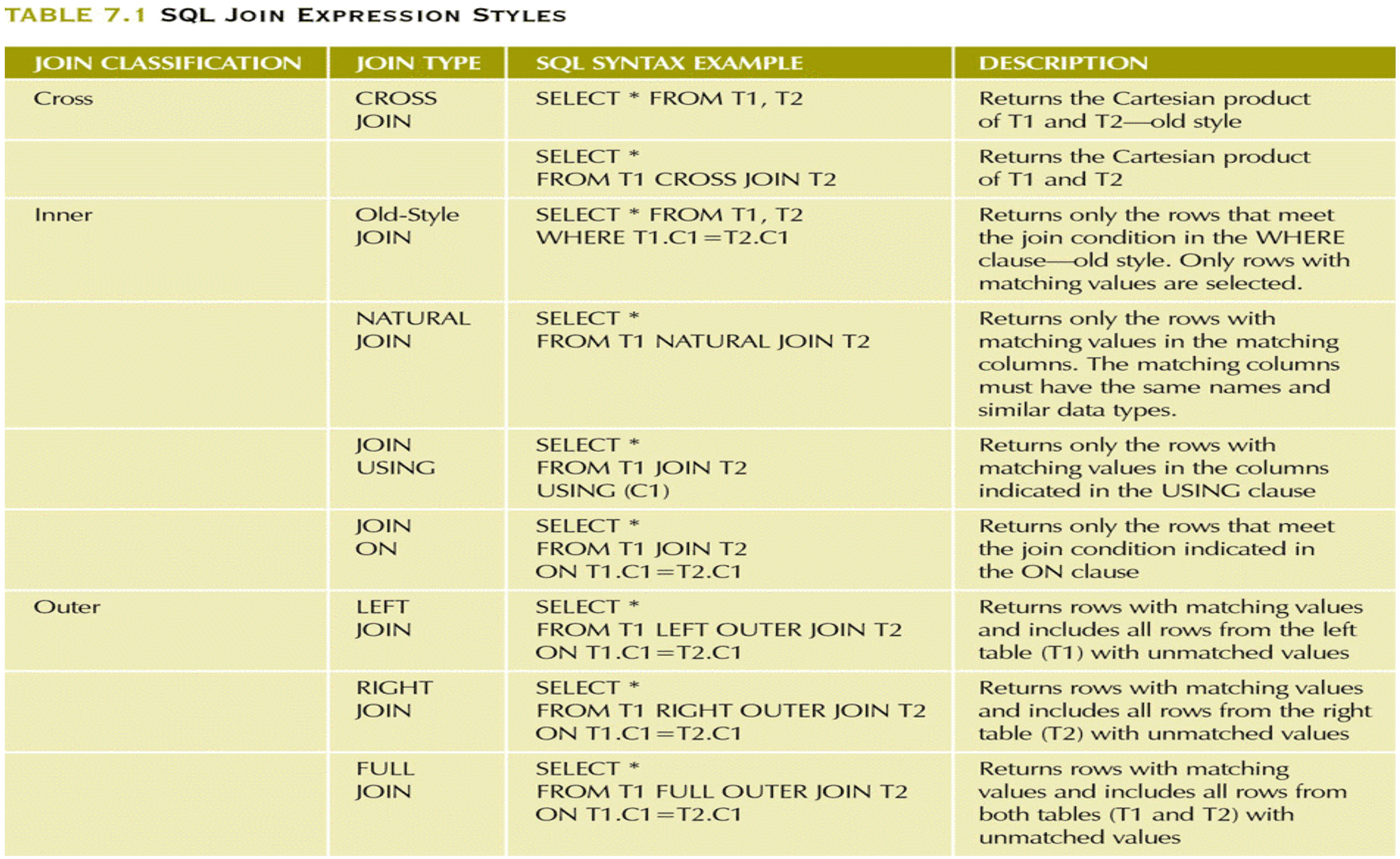
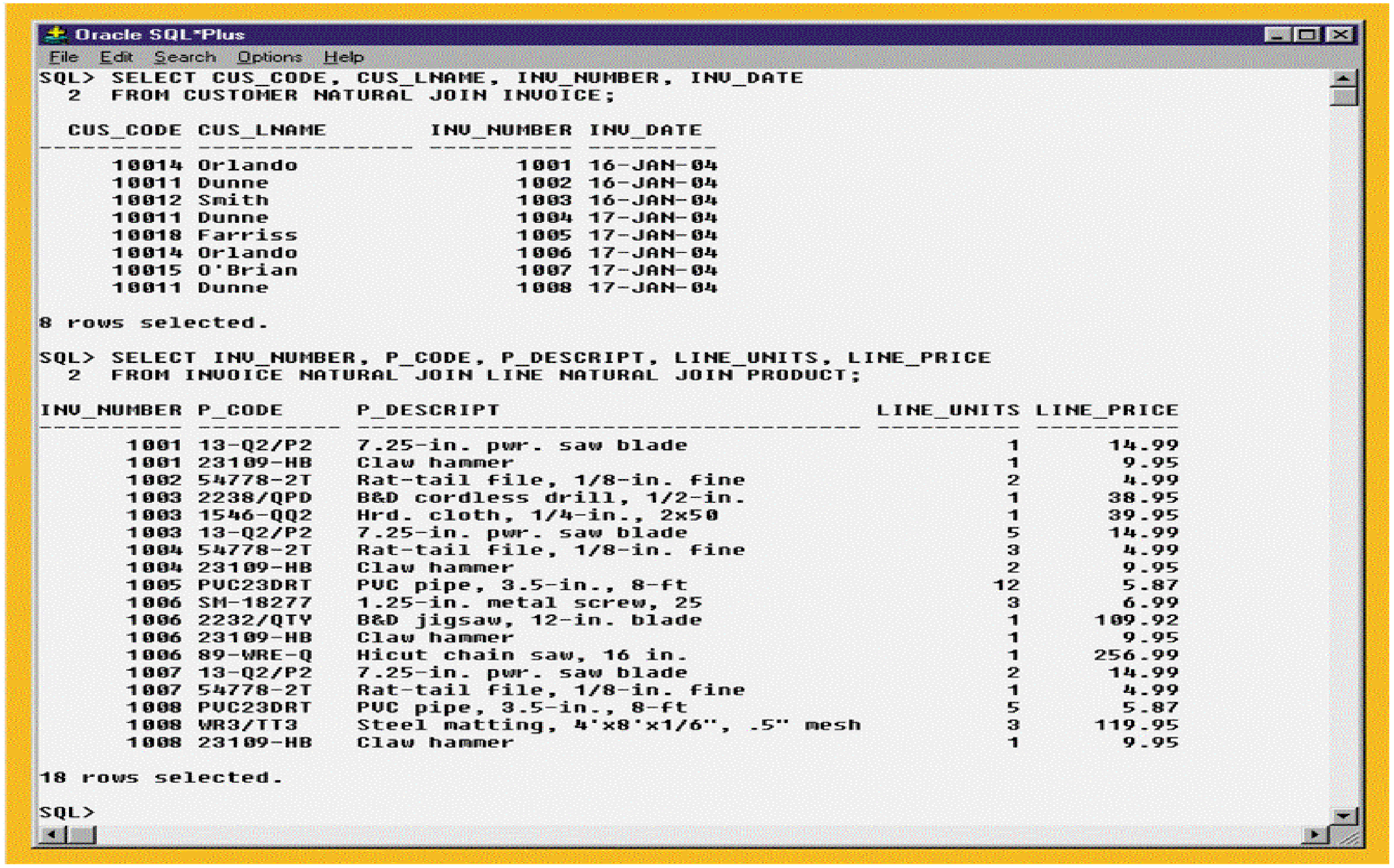


TABLE 7.1 SQL Join Expression Styles

JOIN CLASSIFICATION	JOIN TYPE	SQL SYNTAX EXAMPLE	DESCRIPTION
Cross	CROSS JOIN	SELECT * FROM T1, T2	Returns the Cartesian product of T1 and T2—old style
		SELECT * FROM T1 CROSS JOIN T2	Returns the Cartesian product of T1 and T2
Inner	Old-Style JOIN	SELECT * FROM T1, T2 WHERE T1.C1=T2.C1	Returns only the rows that meet the join condition in the WHERE clause—old style. Only rows with matching values are selected.
	NATURAL JOIN	SELECT * FROM T1 NATURAL JOIN T2	Returns only the rows with matching values in the matching columns. The matching columns must have the same names and similar data types.
	JOIN USING	SELECT * FROM T1 JOIN T2 USING (C1)	Returns only the rows with matching values in the columns indicated in the USING clause
	JOIN ON	SELECT * FROM T1 JOIN T2 ON T1.C1=T2.C1	Returns only the rows that meet the join condition indicated in the ON clause
Outer	LEFT JOIN	SELECT * FROM T1 LEFT OUTER JOIN T2 ON T1.C1=T2.C1	Returns rows with matching values and includes all rows from the left table (T1) with unmatched values
	RIGHT JOIN	SELECT * FROM T1 RIGHT OUTER JOIN T2 ON T1.C1=T2.C1	Returns rows with matching values and includes all rows from the right table (T2) with unmatched values
	FULL JOIN	SELECT * FROM T1 FULL OUTER JOIN T2 ON T1.C1=T2.C1	Returns rows with matching values and includes all rows from both tables (T1 and T2) with unmatched values

NATURAL JOIN Result

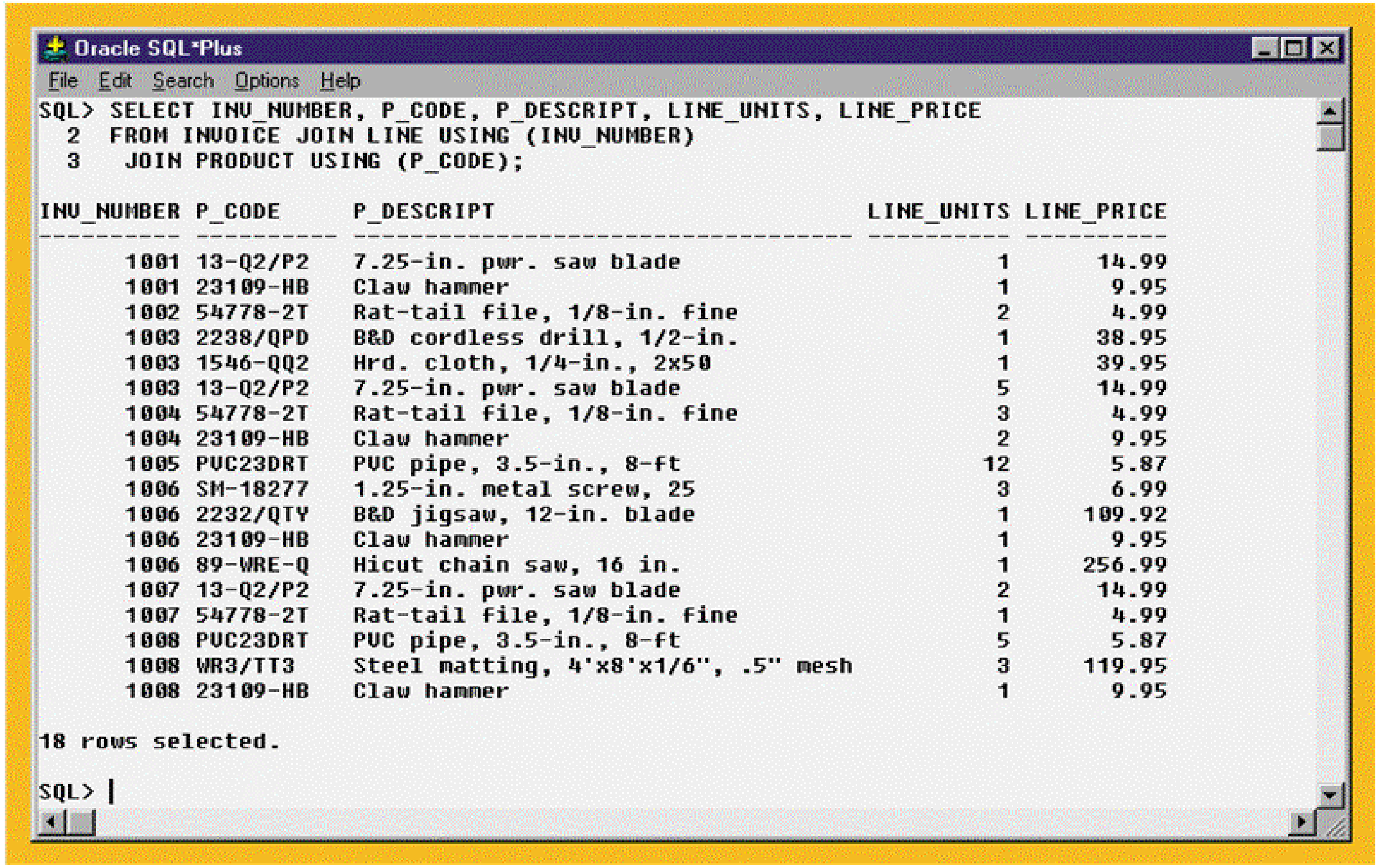
FIGURE 7.7 NATURAL JOIN RESULT



```
SQL> SELECT CUS_CODE, CUS_LNAME, INV_NUMBER, INV_DATE
  2  FROM CUSTOMER NATURAL JOIN INVOICE;

  CUS_CODE CUS_LNAME       INV_NUMBER INV_DATE
---------- --------------- ---------- ---------
     10014 Orlando               1001 16-JAN-04
     10011 Dunne                 1002 16-JAN-04
     10012 Smith                 1003 16-JAN-04
     10011 Dunne                 1004 17-JAN-04
     10018 Farriss               1005 17-JAN-04
     10014 Orlando               1006 17-JAN-04
     10015 O'Brian               1007 17-JAN-04
     10011 Dunne                 1008 17-JAN-04

8 rows selected.

SQL> SELECT INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE
  2  FROM INVOICE NATURAL JOIN LINE NATURAL JOIN PRODUCT;

INV_NUMBER P_CODE     P_DESCRIPT                           LINE_UNITS LINE_PRICE
---------- ---------- ------------------------------------ ---------- ----------
      1001 13-Q2/P2   7.25-in. pwr. saw blade                       1      14.99
      1001 23109-HB   Claw hammer                                   1       9.95
      1002 54778-2T   Rat-tail file, 1/8-in. fine                   2       4.99
      1003 2238/QPD   B&D cordless drill, 1/2-in.                   1      38.95
      1003 1546-QQ2   Hrd. cloth, 1/4-in., 2x50                     1      39.95
      1003 13-Q2/P2   7.25-in. pwr. saw blade                       5      14.99
      1004 54778-2T   Rat-tail file, 1/8-in. fine                   3       4.99
      1004 23109-HB   Claw hammer                                   2       9.95
      1005 PVC23DRT   PVC pipe, 3.5-in., 8-ft                      12       5.87
      1006 SM-18277   1.25-in. metal screw, 25                      3       6.99
      1006 2232/QTY   B&D jigsaw, 12-in. blade                      1     109.92
      1006 23109-HB   Claw hammer                                   1       9.95
      1006 89-WRE-Q   Hicut chain saw, 16 in.                       1     256.99
      1007 13-Q2/P2   7.25-in. pwr. saw blade                       2      14.99
      1007 54778-2T   Rat-tail file, 1/8-in. fine                   1       4.99
      1008 PVC23DRT   PVC pipe, 3.5-in., 8-ft                       5       5.87
      1008 WR3/TT3    Steel matting, 4'x8'x1/6", .5" mesh           3     119.95
      1008 23109-HB   Claw hammer                                   1       9.95

18 rows selected.

SQL>
```

JOIN USING Result

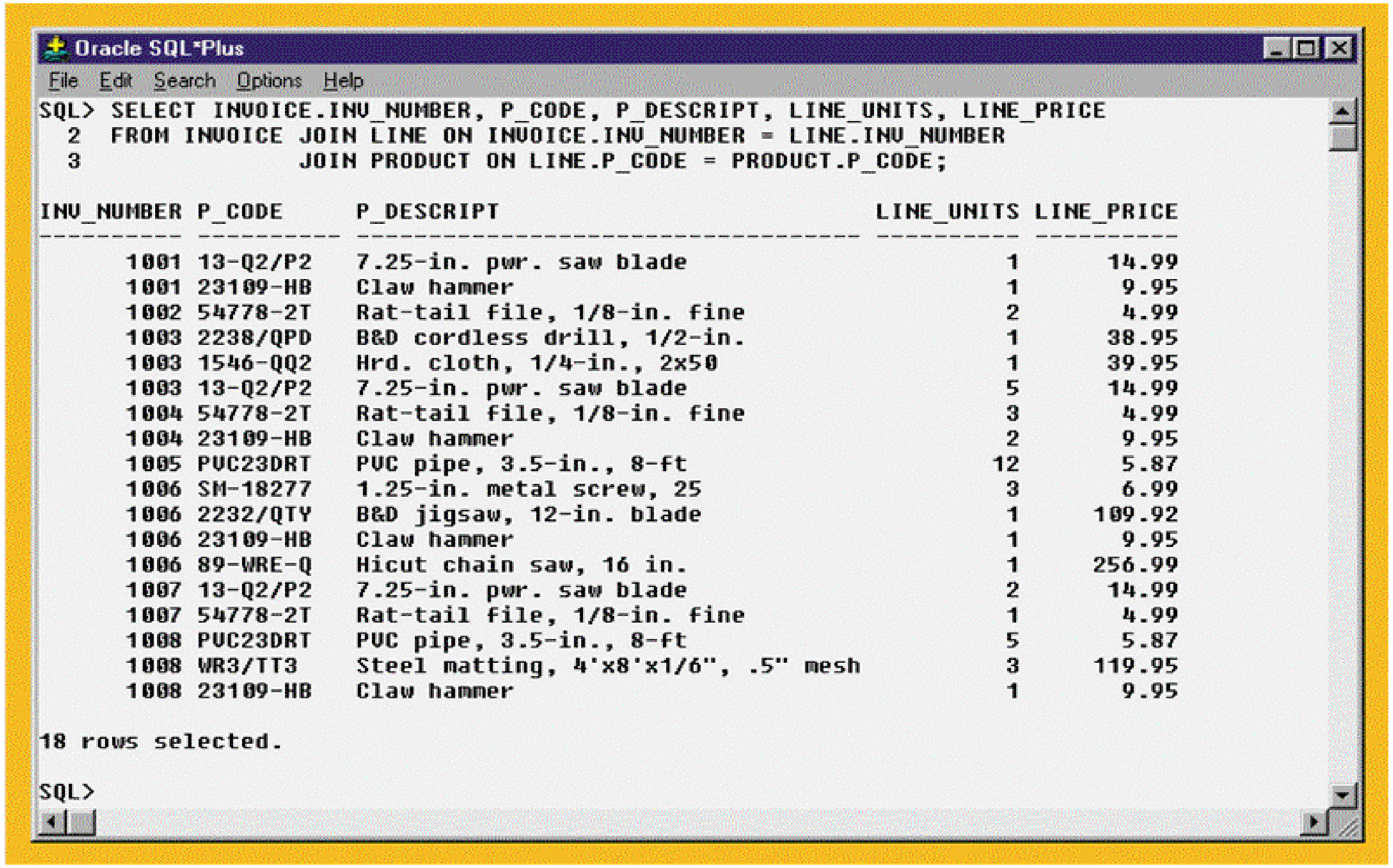
FIGURE 7.8 JOIN USING RESULT



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

INV_NUMBER P_CODE     P_DESCRIPT                          LINE_UNITS LINE_PRICE
---------- ---------- ----------------------------------- ---------- ----------
      1001 13-Q2/P2   7.25-in. pwr. saw blade                      1      14.99
      1001 23109-HB   Claw hammer                                  1       9.95
      1002 54778-2T   Rat-tail file, 1/8-in. fine                  2       4.99
      1003 2238/QPD   B&D cordless drill, 1/2-in.                  1      38.95
      1003 1546-QQ2   Hrd. cloth, 1/4-in., 2x50                    1      39.95
      1003 13-Q2/P2   7.25-in. pwr. saw blade                      5      14.99
      1004 54778-2T   Rat-tail file, 1/8-in. fine                  3       4.99
      1004 23109-HB   Claw hammer                                  2       9.95
      1005 PVC23DRT   PVC pipe, 3.5-in., 8-ft                     12       5.87
      1006 SM-18277   1.25-in. metal screw, 25                     3       6.99
      1006 2232/QTY   B&D jigsaw, 12-in. blade                     1     109.92
      1006 23109-HB   Claw hammer                                  1       9.95
      1006 89-WRE-Q   Hicut chain saw, 16 in.                      1     256.99
      1007 13-Q2/P2   7.25-in. pwr. saw blade                      2      14.99
      1007 54778-2T   Rat-tail file, 1/8-in. fine                  1       4.99
      1008 PVC23DRT   PVC pipe, 3.5-in., 8-ft                      5       5.87
      1008 WR3/TT3    Steel matting, 4'x8'x1/6", .5" mesh          3     119.95
      1008 23109-HB   Claw hammer                                  1       9.95

18 rows selected.

SQL> |
```

JOIN ON Result

FIGURE 7.9 JOIN ON RESULT



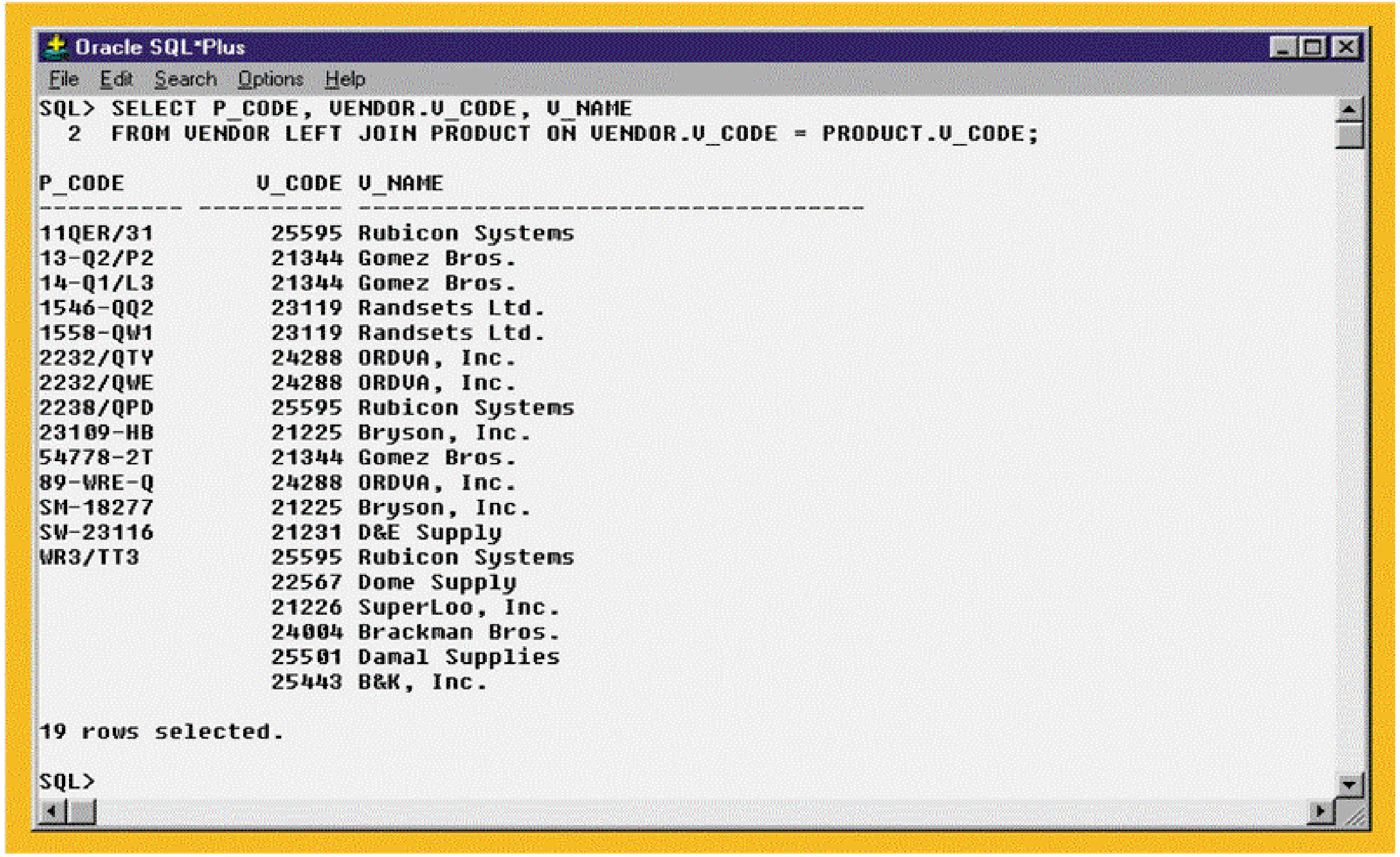
```
SQL> SELECT INVOICE.INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE
  2  FROM INVOICE JOIN LINE ON INVOICE.INV_NUMBER = LINE.INV_NUMBER
  3                JOIN PRODUCT ON LINE.P_CODE = PRODUCT.P_CODE;

INV_NUMBER P_CODE     P_DESCRIPT                           LINE_UNITS LINE_PRICE
---------- ---------- ------------------------------------ ---------- ----------
      1001 13-Q2/P2   7.25-in. pwr. saw blade                       1      14.99
      1001 23109-HB   Claw hammer                                   1       9.95
      1002 54778-2T   Rat-tail file, 1/8-in. fine                   2       4.99
      1003 2238/QPD   B&D cordless drill, 1/2-in.                   1      38.95
      1003 1546-QQ2   Hrd. cloth, 1/4-in., 2x50                     1      39.95
      1003 13-Q2/P2   7.25-in. pwr. saw blade                       5      14.99
      1004 54778-2T   Rat-tail file, 1/8-in. fine                   3       4.99
      1004 23109-HB   Claw hammer                                   2       9.95
      1005 PVC23DRT   PVC pipe, 3.5-in., 8-ft                      12       5.87
      1006 SM-18277   1.25-in. metal screw, 25                      3       6.99
      1006 2232/QTY   B&D jigsaw, 12-in. blade                      1     109.92
      1006 23109-HB   Claw hammer                                   1       9.95
      1006 89-WRE-Q   Hicut chain saw, 16 in.                       1     256.99
      1007 13-Q2/P2   7.25-in. pwr. saw blade                       2      14.99
      1007 54778-2T   Rat-tail file, 1/8-in. fine                   1       4.99
      1008 PVC23DRT   PVC pipe, 3.5-in., 8-ft                       5       5.87
      1008 WR3/TT3    Steel matting, 4'x8'x1/6", .5" mesh           3     119.95
      1008 23109-HB   Claw hammer                                   1       9.95

18 rows selected.

SQL>
```

Outer Joins

- Returns not only rows matching join condition but also rows with unmatched values
- Three types:
  - Left
  - Right
  - Full

LEFT JOIN Result

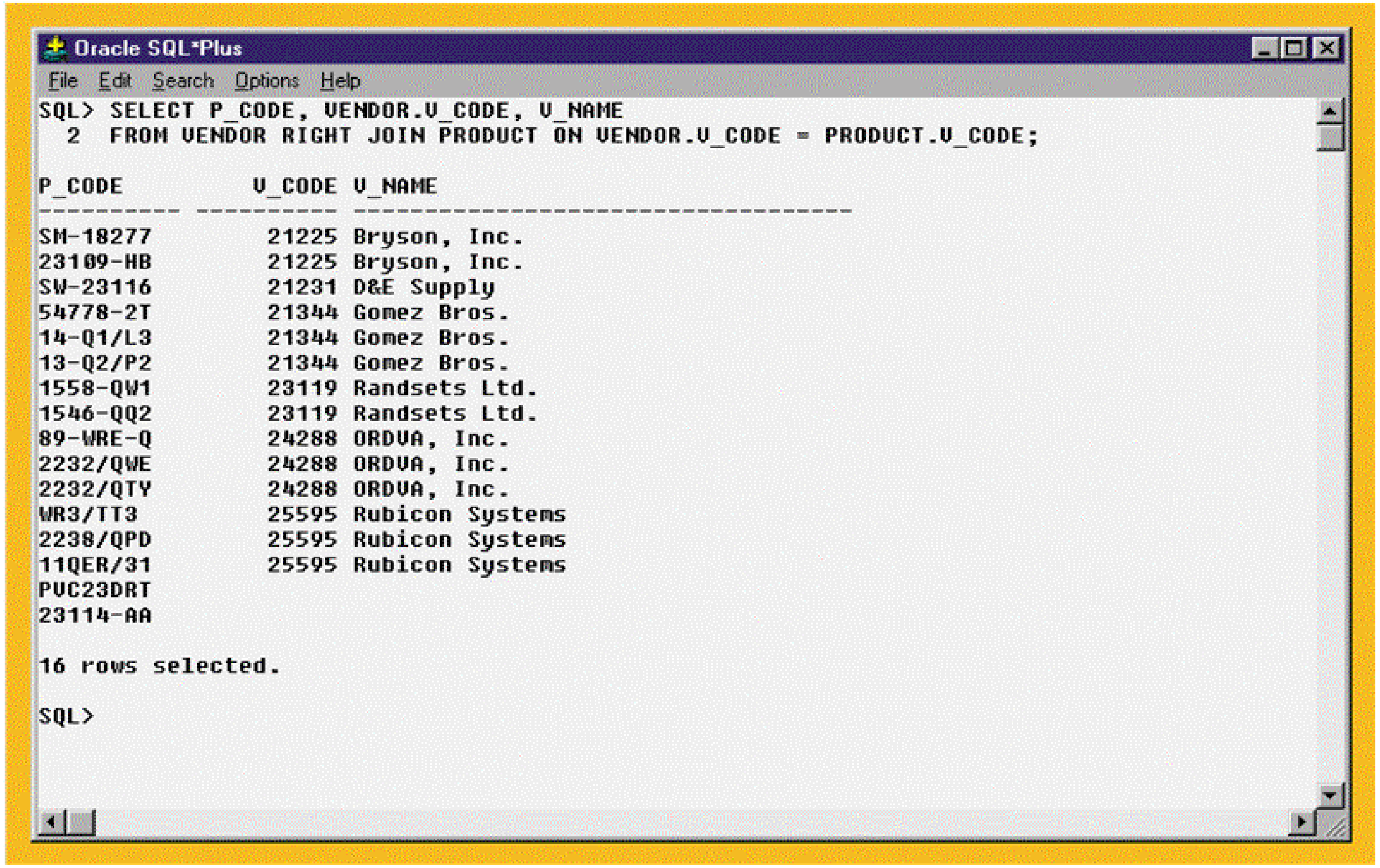
FIGURE 7.10 LEFT JOIN RESULT



```
SQL> SELECT P_CODE, VENDOR.V_CODE, V_NAME
  2  FROM VENDOR LEFT JOIN PRODUCT ON VENDOR.V_CODE = PRODUCT.V_CODE;

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

19 rows selected.

SQL>
```

RIGHT JOIN Result

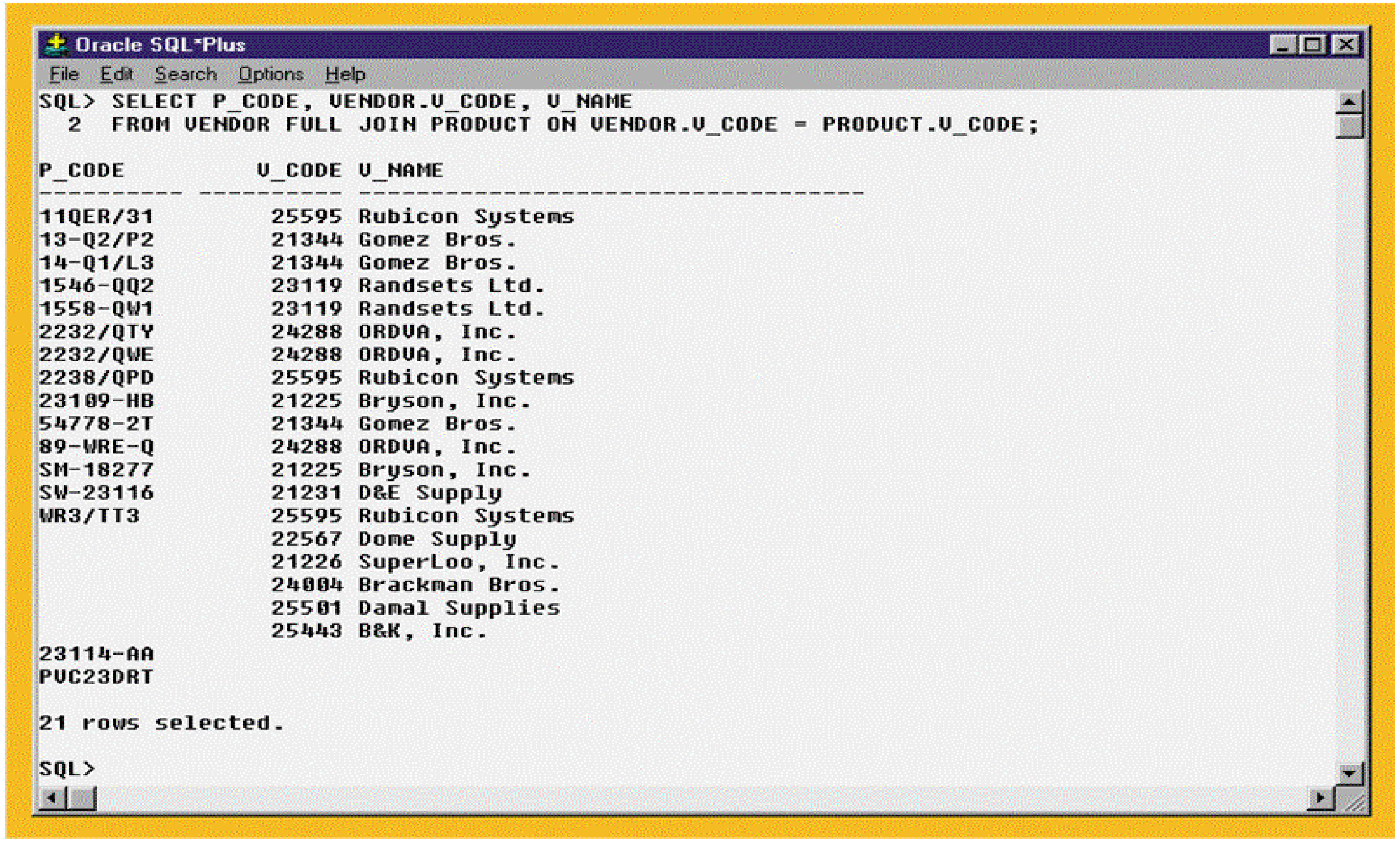
FIGURE 7.11 RIGHT JOIN RESULT



```
SQL> SELECT P_CODE, VENDOR.V_CODE, V_NAME
  2  FROM VENDOR RIGHT JOIN PRODUCT ON VENDOR.V_CODE = PRODUCT.V_CODE;

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

16 rows selected.

SQL>
```

FULL JOIN Result

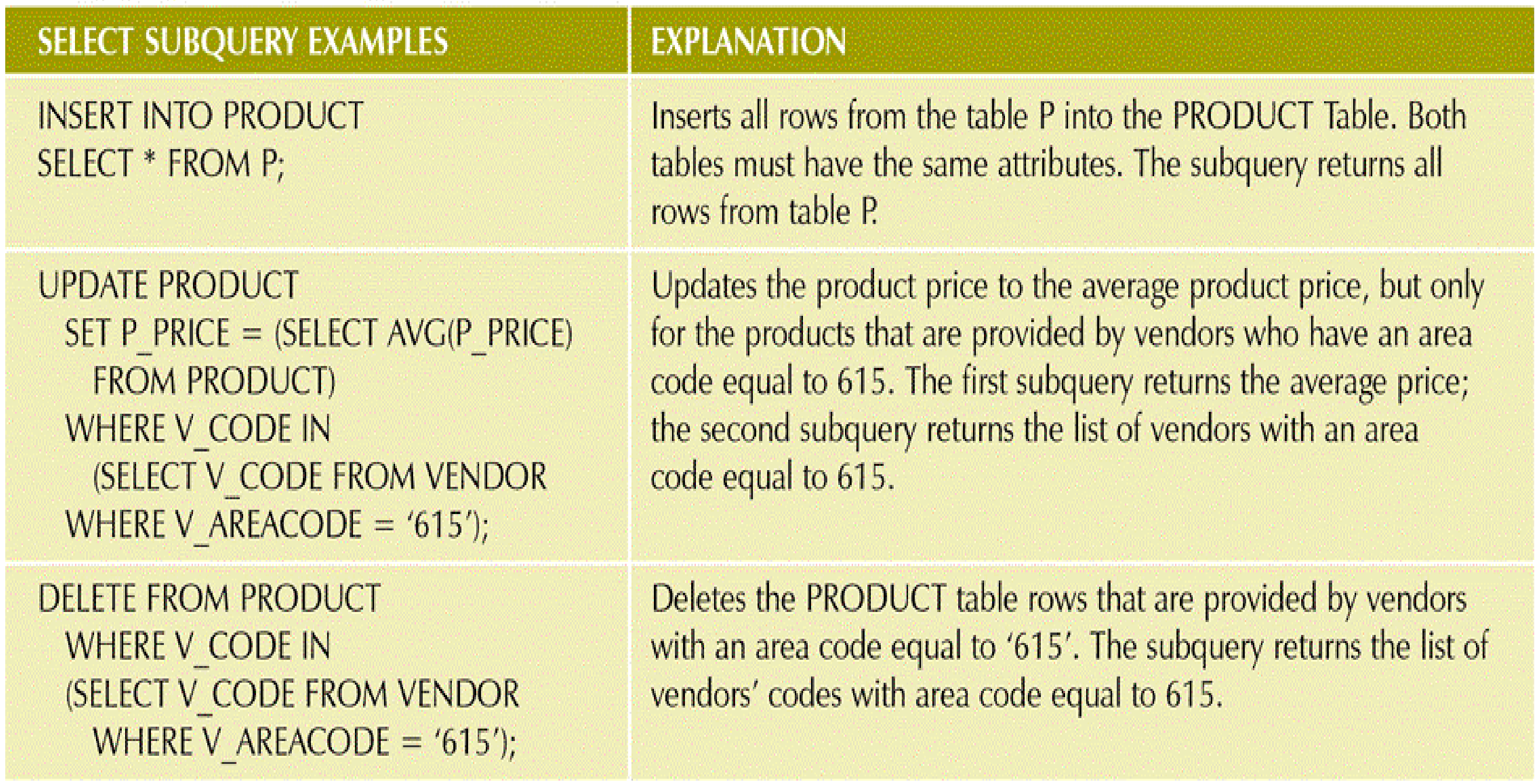
FIGURE 7.12 FULL JOIN RESULT



```
SQL> SELECT P_CODE, VENDOR.V_CODE, V_NAME
  2  FROM VENDOR FULL JOIN PRODUCT ON VENDOR.V_CODE = PRODUCT.V_CODE;

P_CODE         V_CODE V_NAME
---------- ---------- -----------------------------------
11QER/31        25595 Rubicon Systems
13-Q2/P2        21344 Gomez Bros.
14-Q1/L3        21344 Gomez Bros.
1546-QQ2        23119 Randsets Ltd.
1558-QW1        23119 Randsets Ltd.
2232/QTY        24288 ORDVA, Inc.
2232/QWE        24288 ORDVA, Inc.
2238/QPD        25595 Rubicon Systems
23109-HB        21225 Bryson, Inc.
54778-2T        21344 Gomez Bros.
89-WRE-Q        24288 ORDVA, Inc.
SM-18277        21225 Bryson, Inc.
SW-23116        21231 D&E Supply
WR3/TT3         25595 Rubicon Systems
                22567 Dome Supply
                21226 SuperLoo, Inc.
                24004 Brackman Bros.
                25501 Damal Supplies
                25443 B&K, Inc.
23114-AA
PVC23DRT

21 rows selected.

SQL>
```

SELECT Subquery Examples

TABLE 7.2 SELECT SUBQUERY EXAMPLES



SELECT SUBQUERY EXAMPLES	EXPLANATION

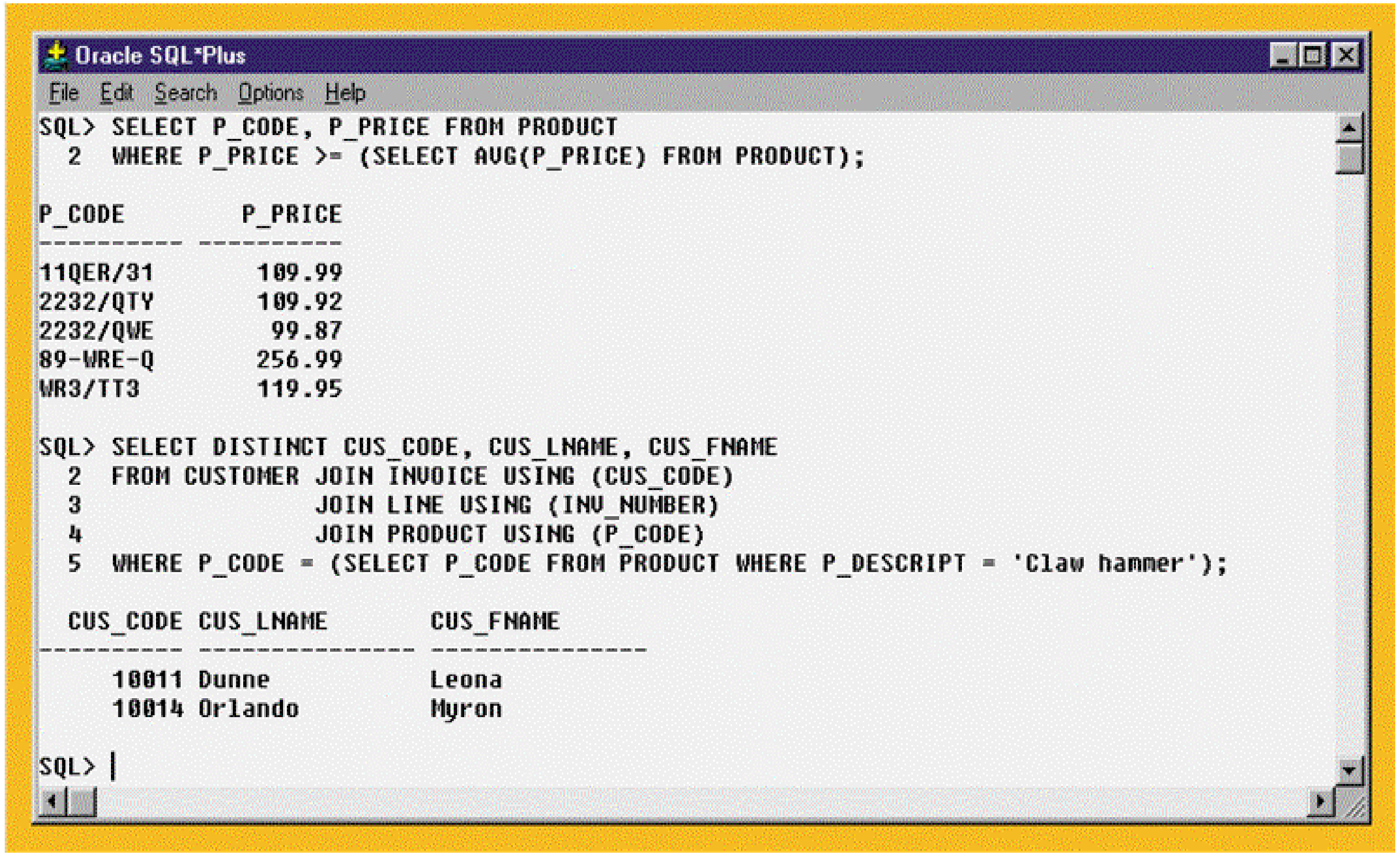
INSERT INTO PRODUCT
SELECT * FROM P;
	Inserts all rows from the table P into the PRODUCT Table. Both tables must have the same attributes. The subquery returns all rows from table P.

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 an area code equal to '615'. The subquery returns the list of vendors' codes with area code equal to 615.

WHERE Subquery Examples

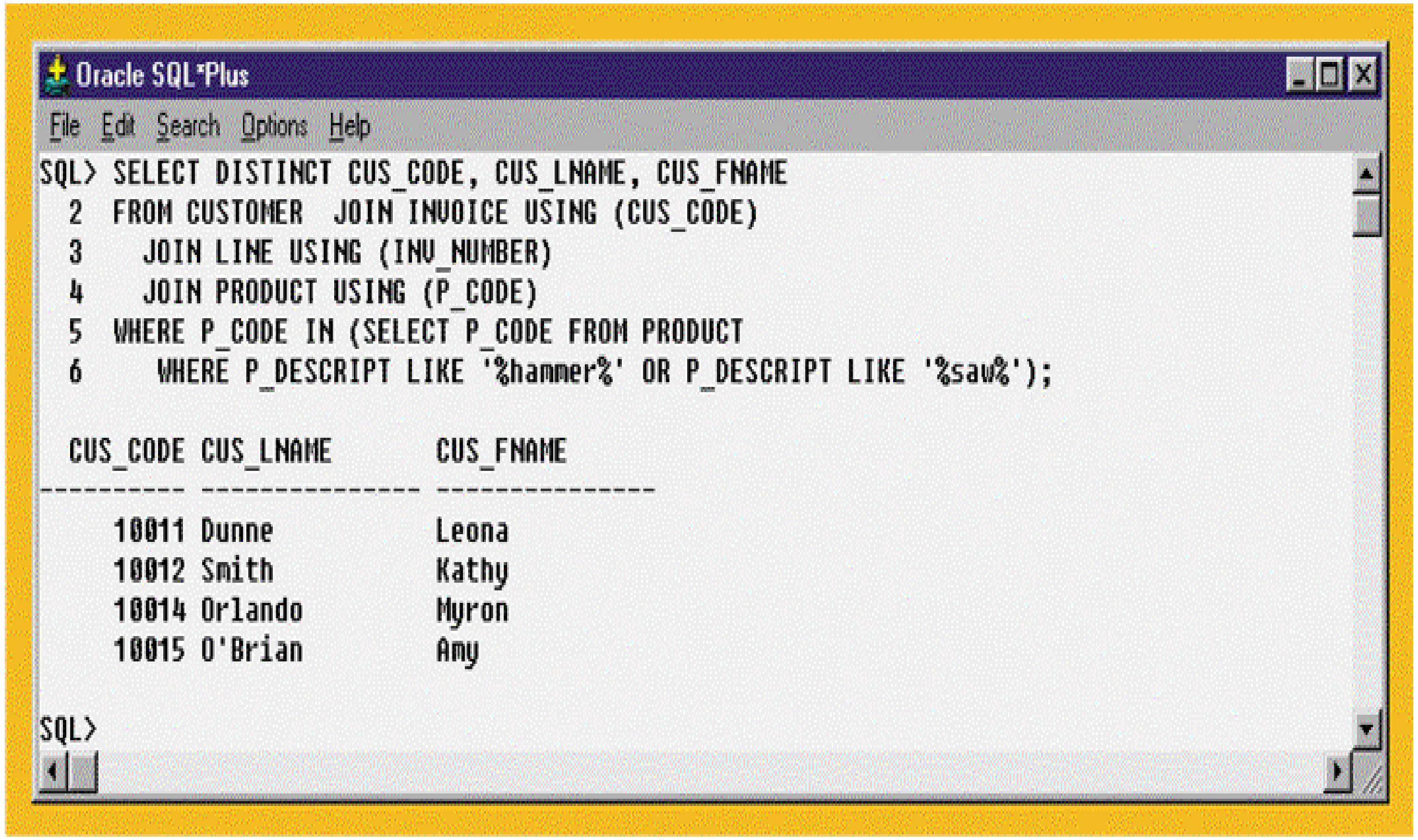
FIGURE 7.13 WHERE SUBQUERY EXAMPLES



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

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

SQL> SELECT DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
  2  FROM CUSTOMER JOIN INVOICE USING (CUS_CODE)
  3                JOIN LINE USING (INV_NUMBER)
  4                JOIN PRODUCT USING (P_CODE)
  5  WHERE P_CODE = (SELECT P_CODE FROM PRODUCT WHERE P_DESCRIPT = 'Claw hammer');

  CUS_CODE CUS_LNAME       CUS_FNAME
---------- --------------- ---------------
     10011 Dunne           Leona
     10014 Orlando         Myron

SQL> |
```

IN Subquery Example

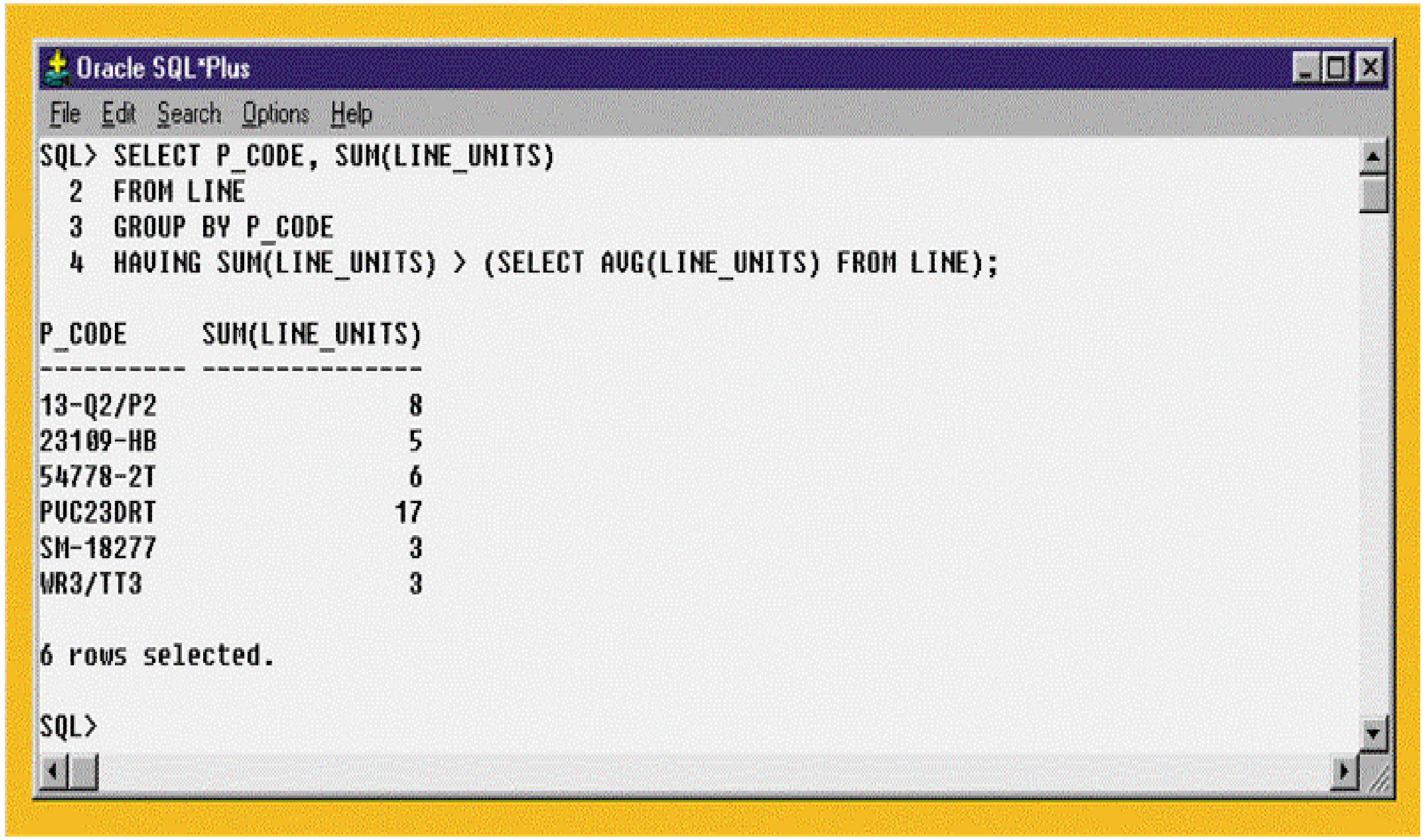
FIGURE 7.14 IN SUBQUERY EXAMPLE



```
SQL> SELECT DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
  FROM CUSTOMER  JOIN INVOICE USING (CUS_CODE)
    JOIN LINE USING (INV_NUMBER)
    JOIN PRODUCT USING (P_CODE)
  WHERE P_CODE IN (SELECT P_CODE FROM PRODUCT
     WHERE P_DESCRIPT LIKE '%hammer%' OR P_DESCRIPT LIKE '%saw%');

  CUS_CODE CUS_LNAME       CUS_FNAME
---------- --------------- ---------------
     10011 Dunne           Leona
     10012 Smith           Kathy
     10014 Orlando         Myron
     10015 O'Brian         Amy

SQL>
```

HAVING Subquery Example

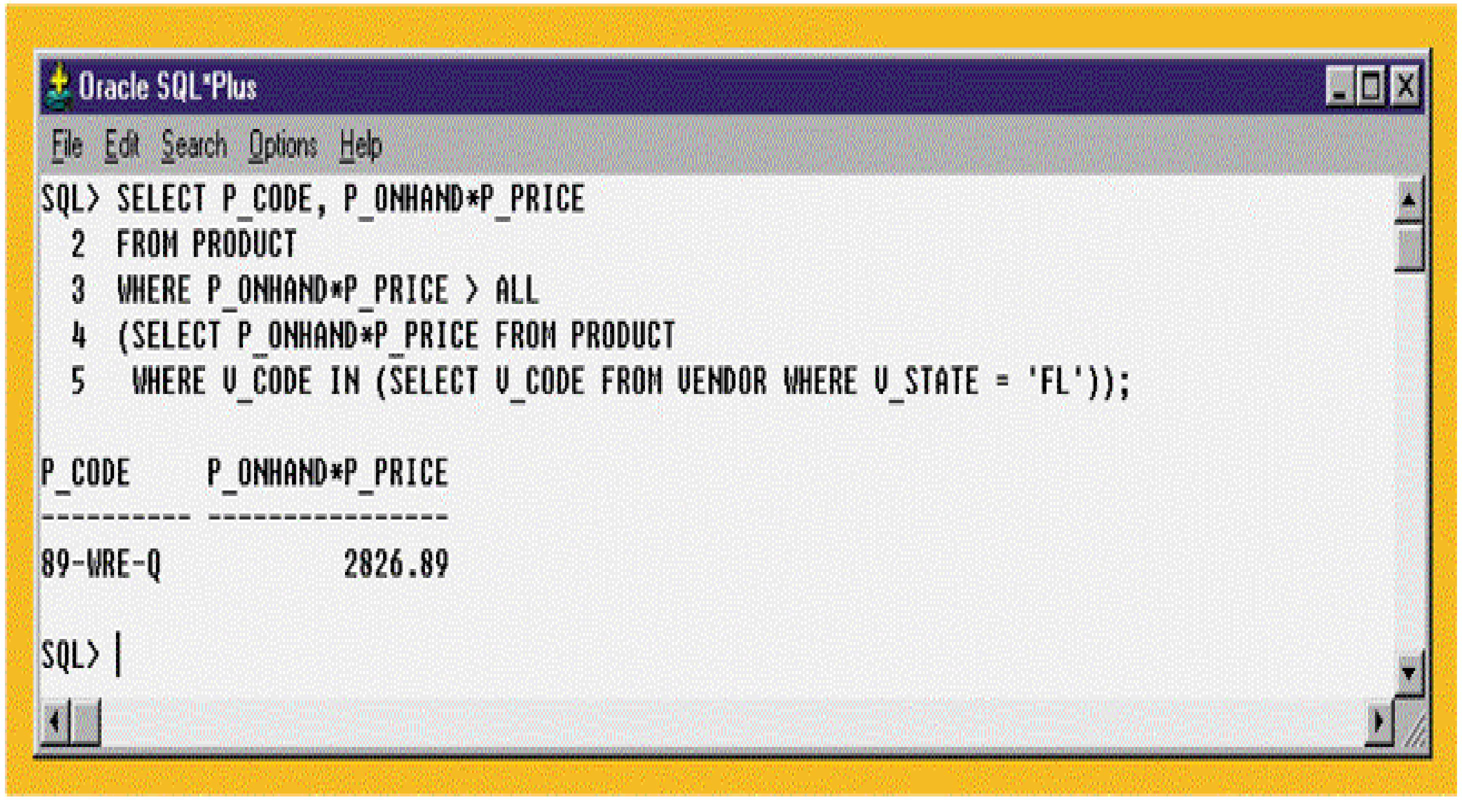
FIGURE 7.15 HAVING SUBQUERY EXAMPLE



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

P_CODE     SUM(LINE_UNITS)
---------- ---------------
13-Q2/P2                 8
23109-HB                 5
54778-2T                 6
PVC23DRT                17
SM-18277                 3
WR3/TT3                  3

6 rows selected.

SQL>
```

Multirow Subquery Operator Example

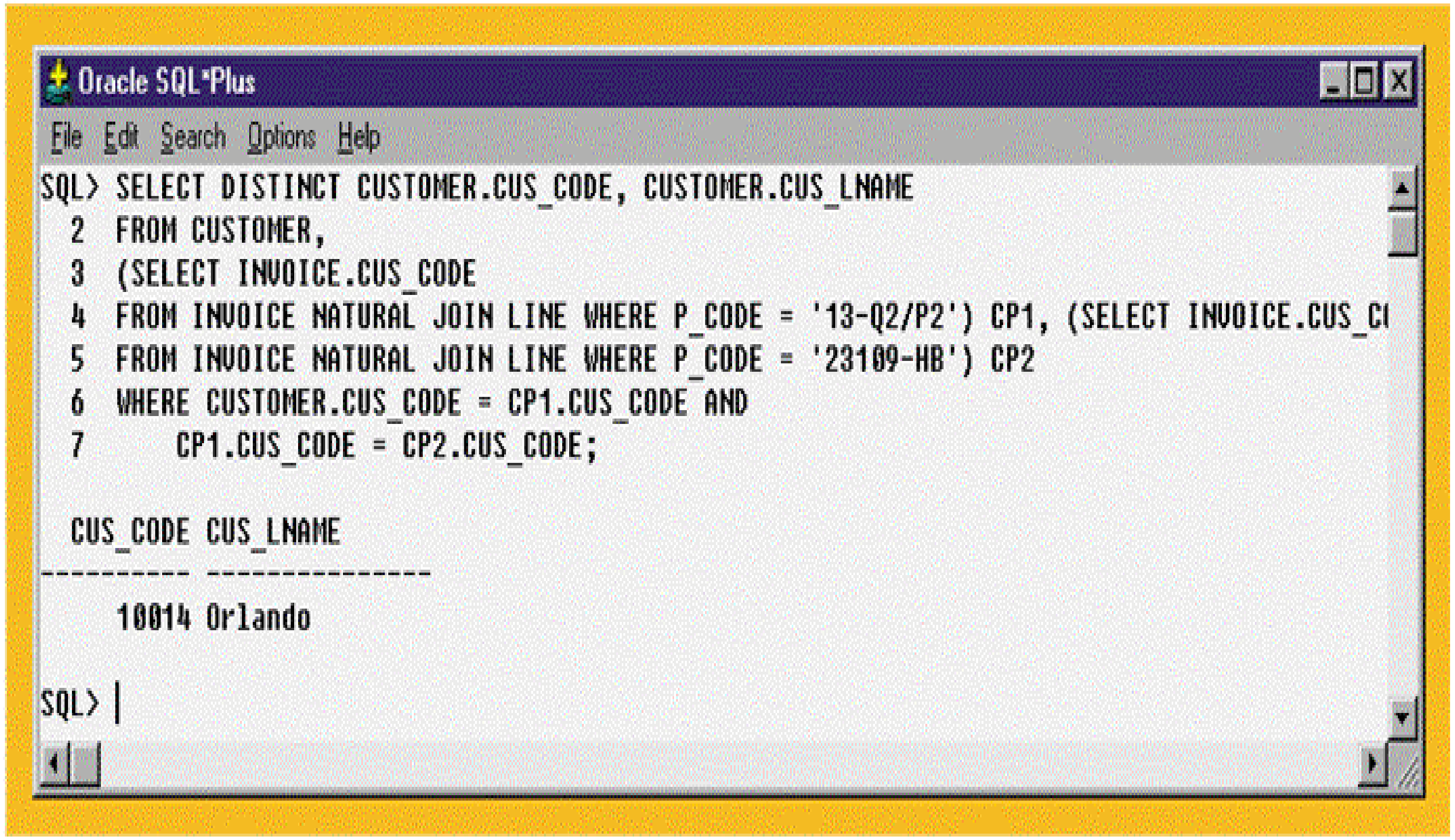
FIGURE 7.16 MULTIROW SUBQUERY OPERATOR EXAMPLE



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

P_CODE     P_ONHAND*P_PRICE
---------- ----------------
89-WRE-Q            2826.89

SQL> |
```

FROM Subquery Example

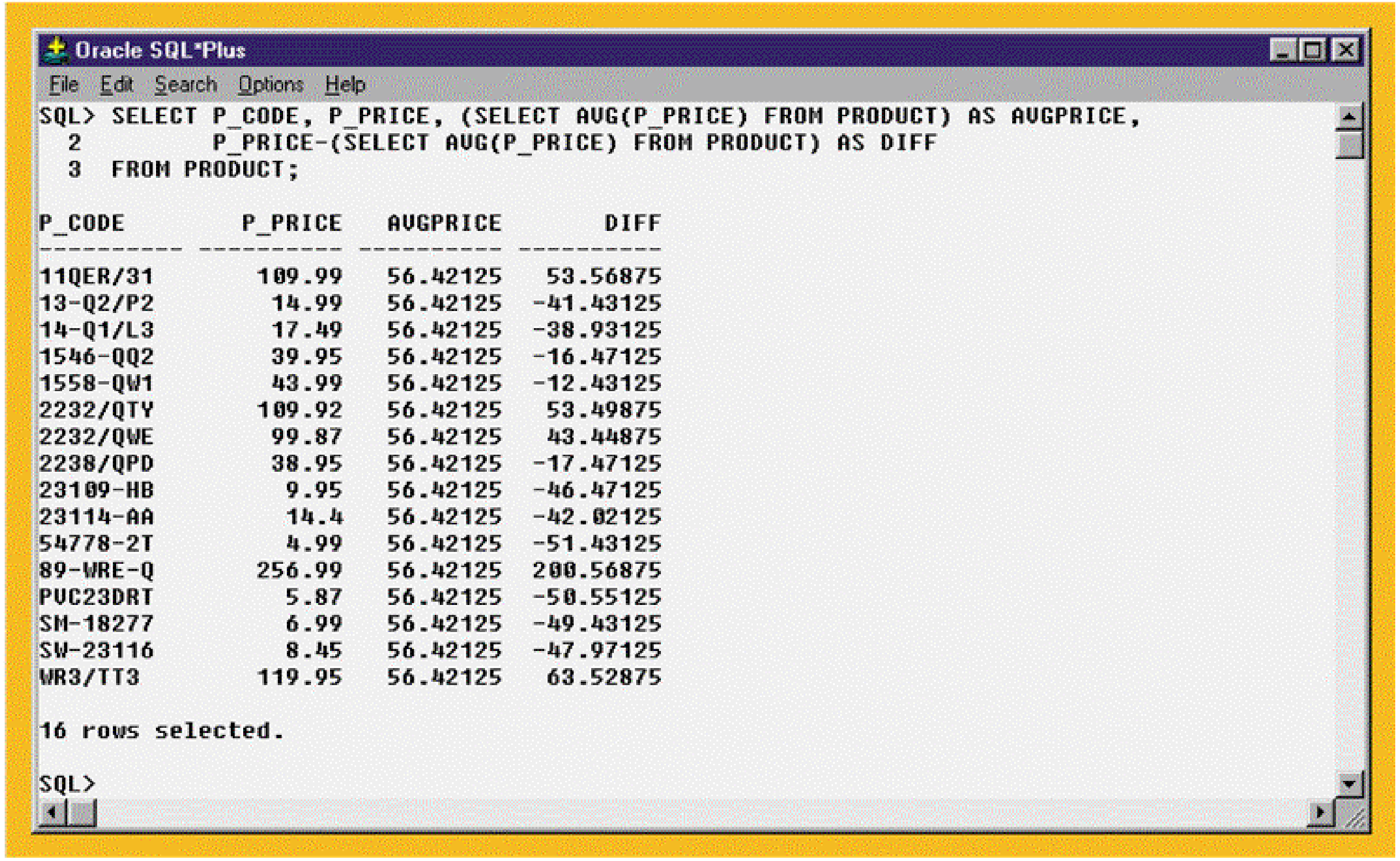
FIGURE 7.17 FROM SUBQUERY EXAMPLE



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

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

SQL>
```

Inline Subquery Example

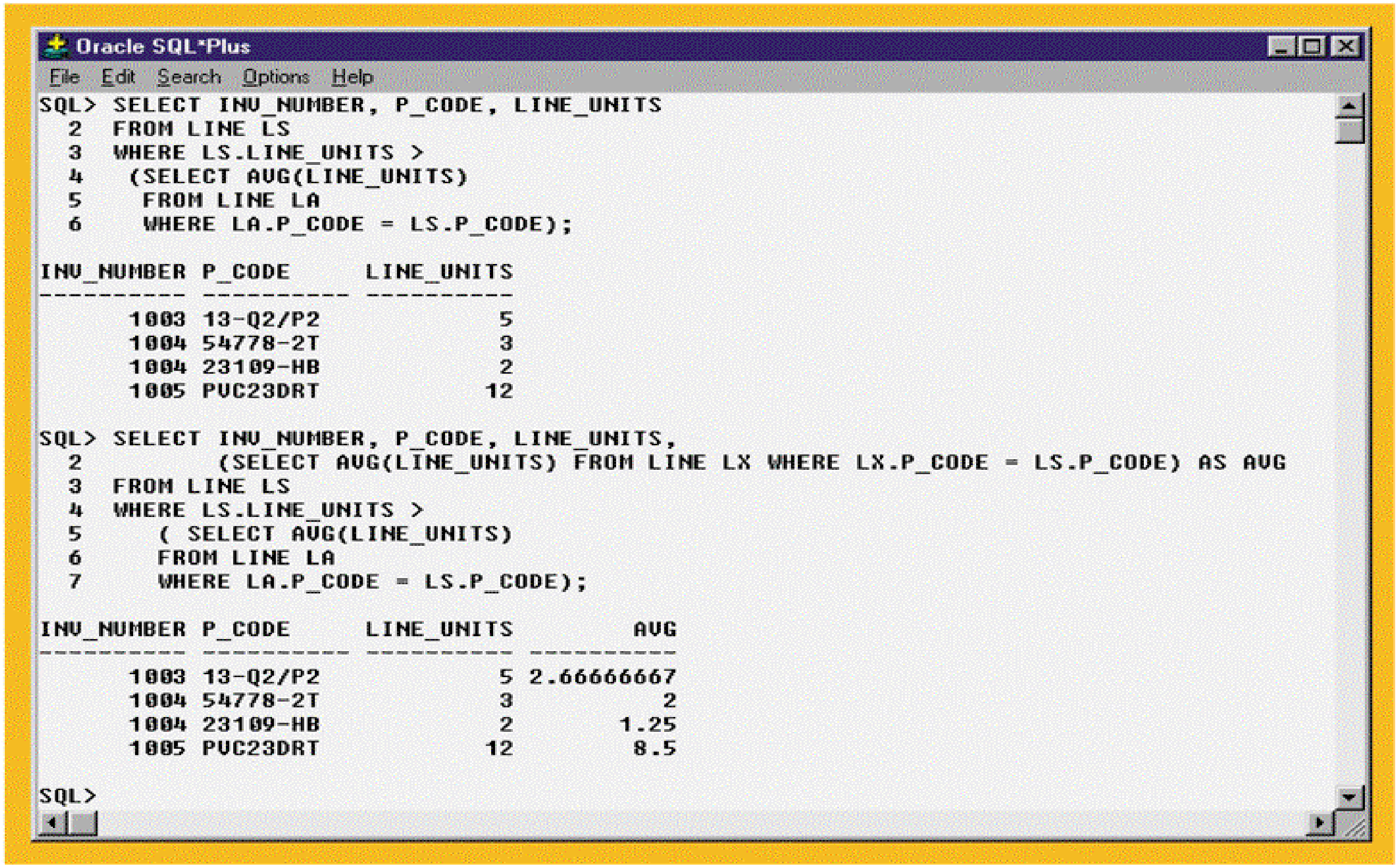
FIGURE 7.18 INLINE SUBQUERY EXAMPLE



```
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>
```

Correlated Subquery Examples

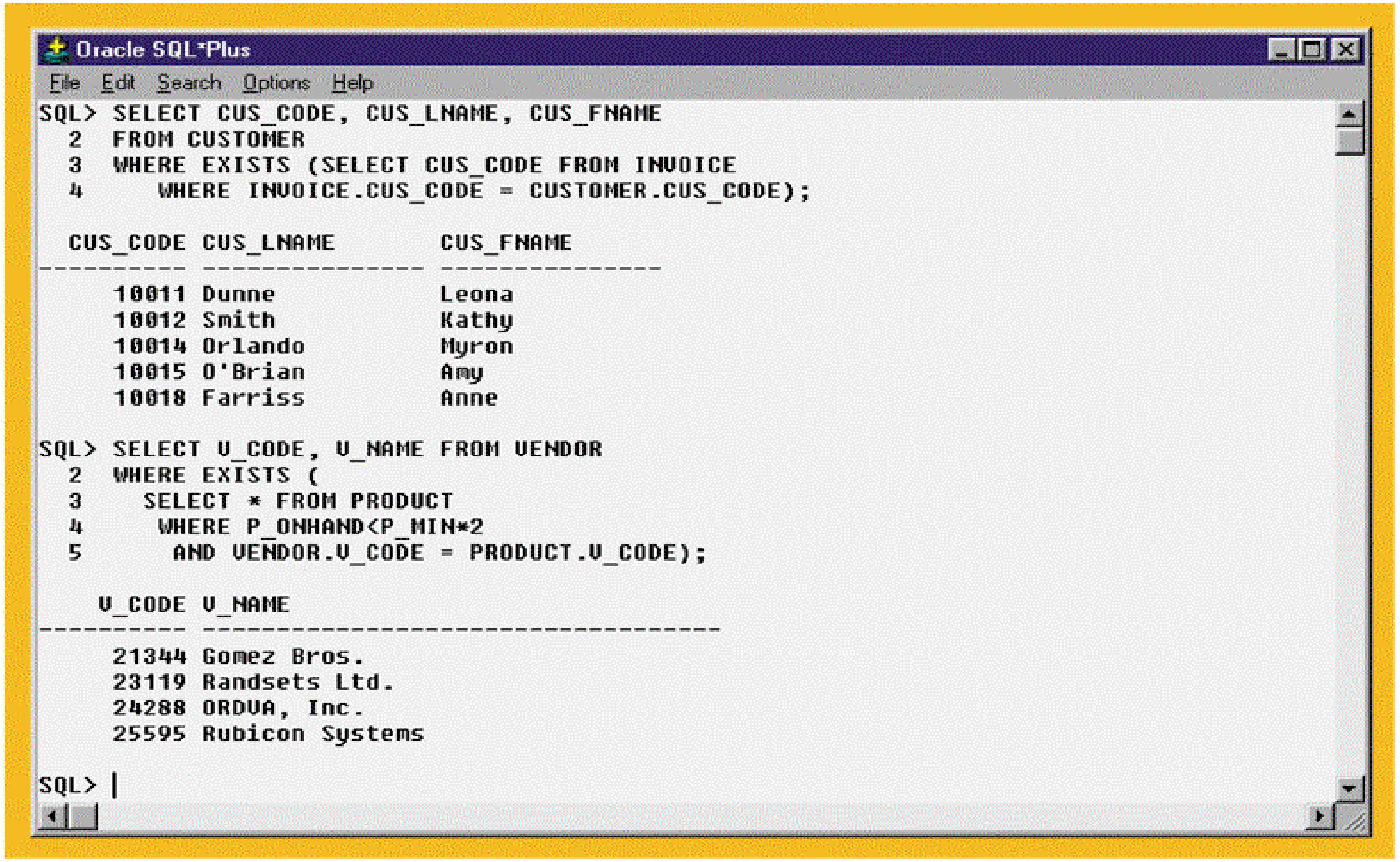
FIGURE 7.20 CORRELATED SUBQUERY EXAMPLES



```
SQL> SELECT INV_NUMBER, P_CODE, LINE_UNITS
  2  FROM LINE LS
  3  WHERE LS.LINE_UNITS >
  4   (SELECT AVG(LINE_UNITS)
  5    FROM LINE LA
  6    WHERE LA.P_CODE = LS.P_CODE);

INV_NUMBER P_CODE     LINE_UNITS
---------- ---------- ----------
      1003 13-Q2/P2            5
      1004 54778-2T            3
      1004 23109-HB            2
      1005 PVC23DRT           12

SQL> SELECT INV_NUMBER, P_CODE, LINE_UNITS,
  2         (SELECT AVG(LINE_UNITS) FROM LINE LX WHERE LX.P_CODE = LS.P_CODE) AS AVG
  3  FROM LINE LS
  4  WHERE LS.LINE_UNITS >
  5     ( SELECT AVG(LINE_UNITS)
  6     FROM LINE LA
  7     WHERE LA.P_CODE = LS.P_CODE);

INV_NUMBER P_CODE     LINE_UNITS        AVG
---------- ---------- ---------- ----------
      1003 13-Q2/P2            5 2.66666667
      1004 54778-2T            3          2
      1004 23109-HB            2       1.25
      1005 PVC23DRT           12        8.5

SQL>
```

EXISTS Correlated Subquery Examples

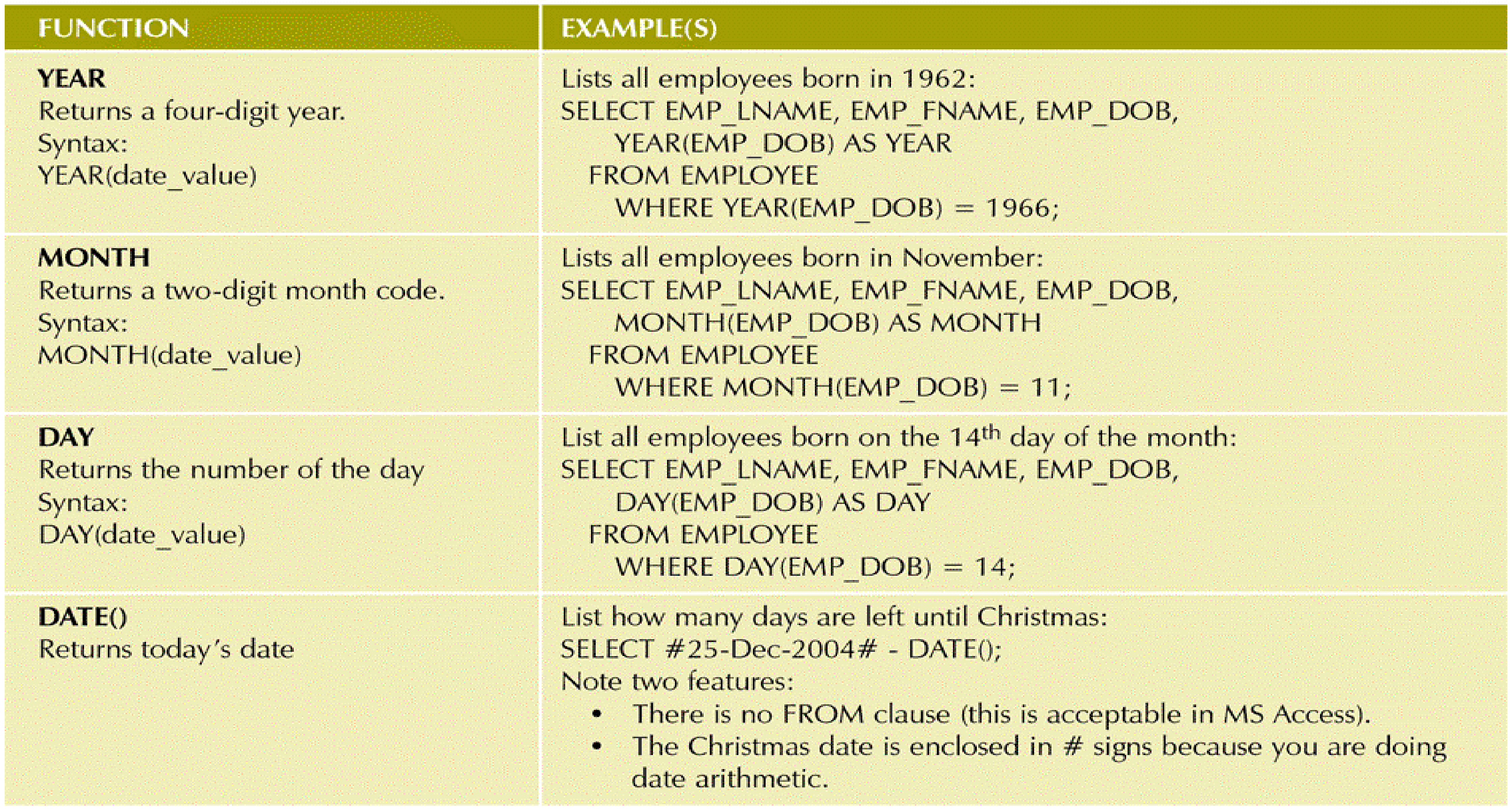
FIGURE 7.21 EXISTS CORRELATED SUBQUERY EXAMPLES



```
SQL> SELECT CUS_CODE, CUS_LNAME, CUS_FNAME
  2  FROM CUSTOMER
  3  WHERE EXISTS (SELECT CUS_CODE FROM INVOICE
  4     WHERE INVOICE.CUS_CODE = CUSTOMER.CUS_CODE);

  CUS_CODE CUS_LNAME       CUS_FNAME
---------- --------------- ---------------
     10011 Dunne           Leona
     10012 Smith           Kathy
     10014 Orlando         Myron
     10015 O'Brian         Amy
     10018 Farriss         Anne

SQL> SELECT V_CODE, V_NAME FROM VENDOR
  2  WHERE EXISTS (
  3    SELECT * FROM PRODUCT
  4     WHERE P_ONHAND<P_MIN*2
  5      AND VENDOR.V_CODE = PRODUCT.V_CODE);

    V_CODE V_NAME
---------- -----------------------------------
     21344 Gomez Bros.
     23119 Randsets Ltd.
     24288 ORDVA, Inc.
     25595 Rubicon Systems

SQL> |
```

Selected MS Access/SQL Server Date/Time Functions

TABLE 7.3 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 1962:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	YEAR(EMP_DOB) AS YEAR
	FROM EMPLOYEE
	WHERE YEAR(EMP_DOB) = 1966;

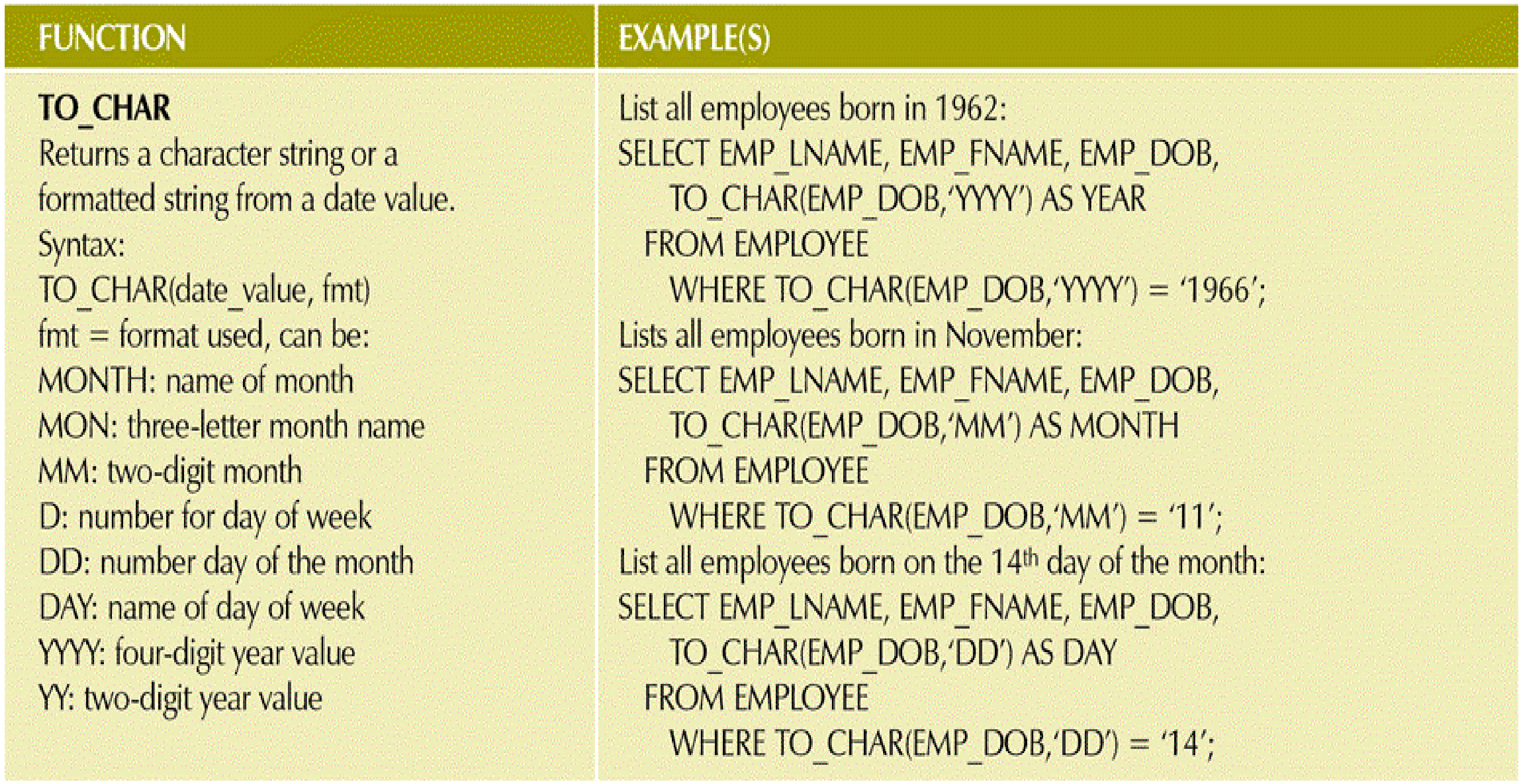
MONTH
Returns a two-digit month code.
Syntax:
MONTH(date_value)
	Lists all employees born in November:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	MONTH(EMP_DOB) AS MONTH
	FROM EMPLOYEE
	WHERE MONTH(EMP_DOB) = 11;

DAY
Returns the number of the day
Syntax:
DAY(date_value)
	List 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	List how many days are left until Christmas: SELECT #25-Dec-2004# - DATE(); Note two features: • There is no FROM clause (this is acceptable in MS Access). • The Christmas date is enclosed in # signs because you are doing date arithmetic.

Selected Oracle Date/Time Functions

TABLE 7.4 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
D: number for day of week
DD: number day of the month
DAY: name of day of week
YYYY: four-digit year value
YY: two-digit year value
	List all employees born in 1962:
	SELECT EMP_LNAME, EMP_FNAME, EMP_DOB,
	TO_CHAR(EMP_DOB,'YYYY') AS YEAR
	FROM EMPLOYEE
	WHERE TO_CHAR(EMP_DOB,'YYYY') = '1966';
	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';
	List 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';

Selected Oracle Date/Time Functions (continued)

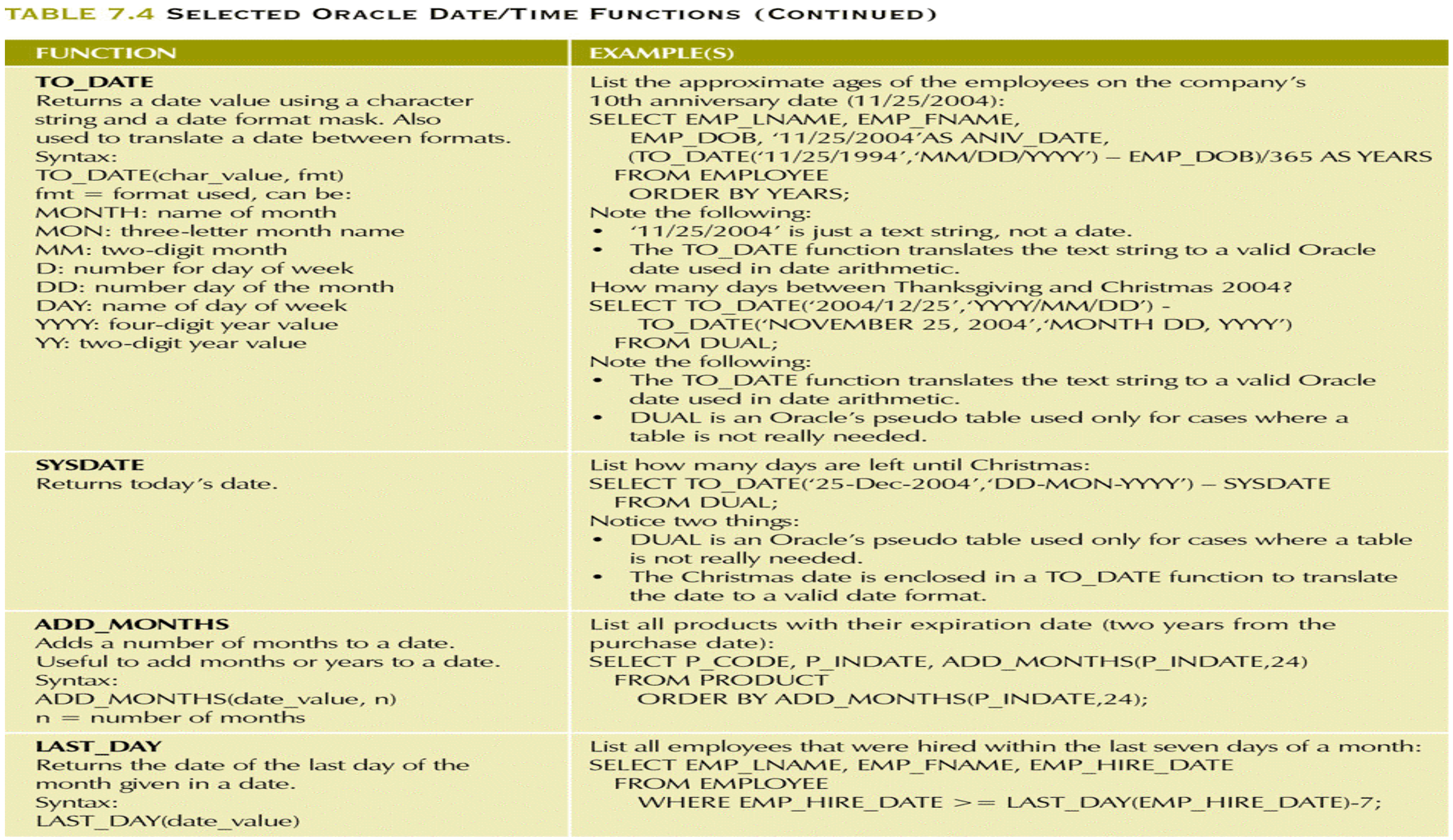


TABLE 7.4 SELECTED ORACLE DATE/TIME FUNCTIONS (CONTINUED)

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 D: number for day of week DD: number day of the month DAY: name of day of week YYYY: four-digit year value YY: two-digit year value	List the approximate ages of the employees on the company's 10th anniversary date (11/25/2004): SELECT EMP_LNAME, EMP_FNAME, EMP_DOB, '11/25/2004'AS ANIV_DATE, (TO_DATE('11/25/1994','MM/DD/YYYY') – EMP_DOB)/365 AS YEARS FROM EMPLOYEE ORDER BY YEARS; Note the following: • '11/25/2004' is just 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 2004? SELECT TO_DATE('2004/12/25','YYYY/MM/DD') - TO_DATE('NOVEMBER 25, 2004','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 an Oracle's pseudo table used only for cases where a table is not really needed.
SYSDATE Returns today's date.	List how many days are left until Christmas: SELECT TO_DATE('25-Dec-2004','DD-MON-YYYY') – SYSDATE FROM DUAL; Notice two things: • DUAL is an 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 to add months or years to a date.
Syntax:
ADD_MONTHS(date_value, n)
n = number of months
	List 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)
	List all employees that 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;

Selected Oracle Numeric Functions

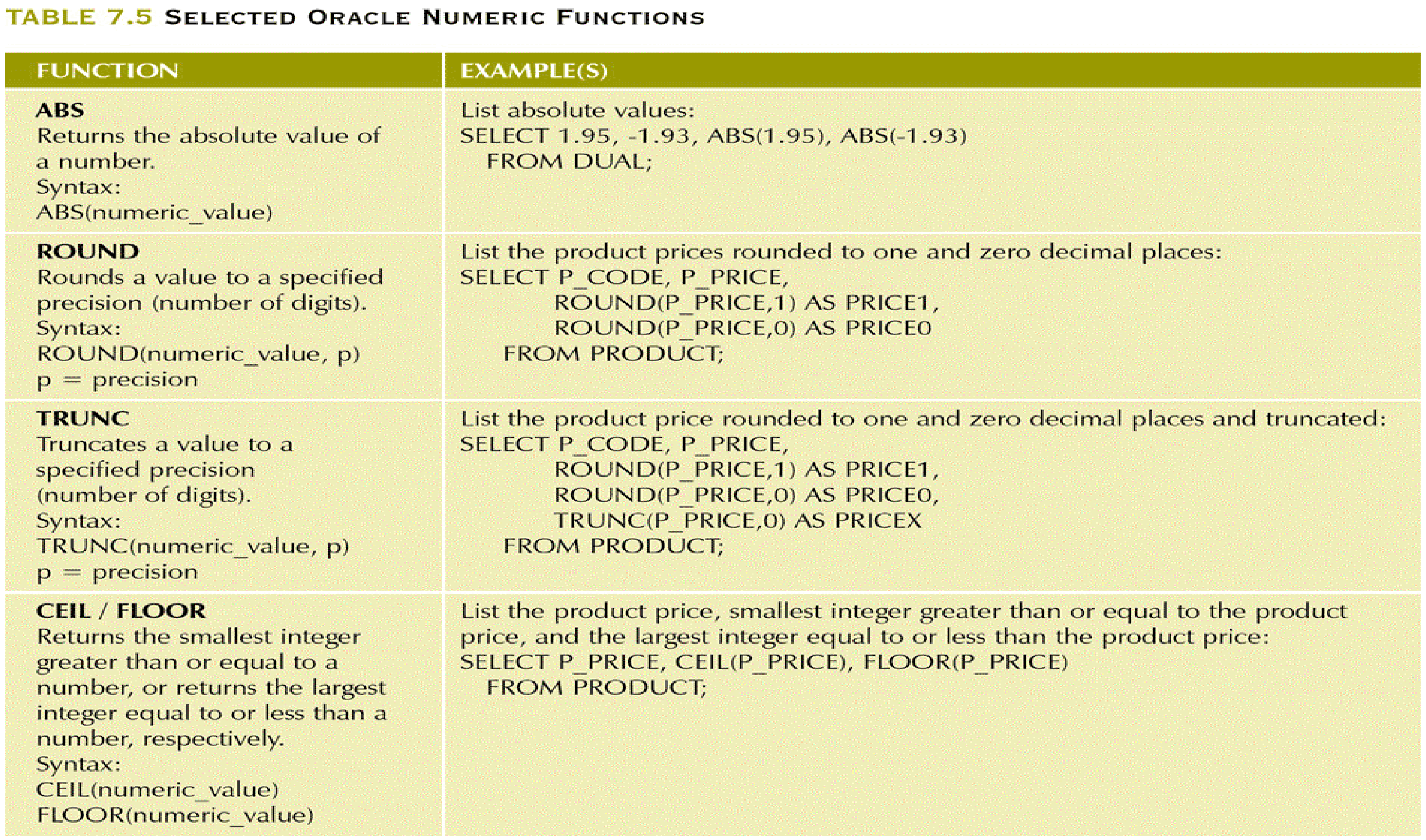
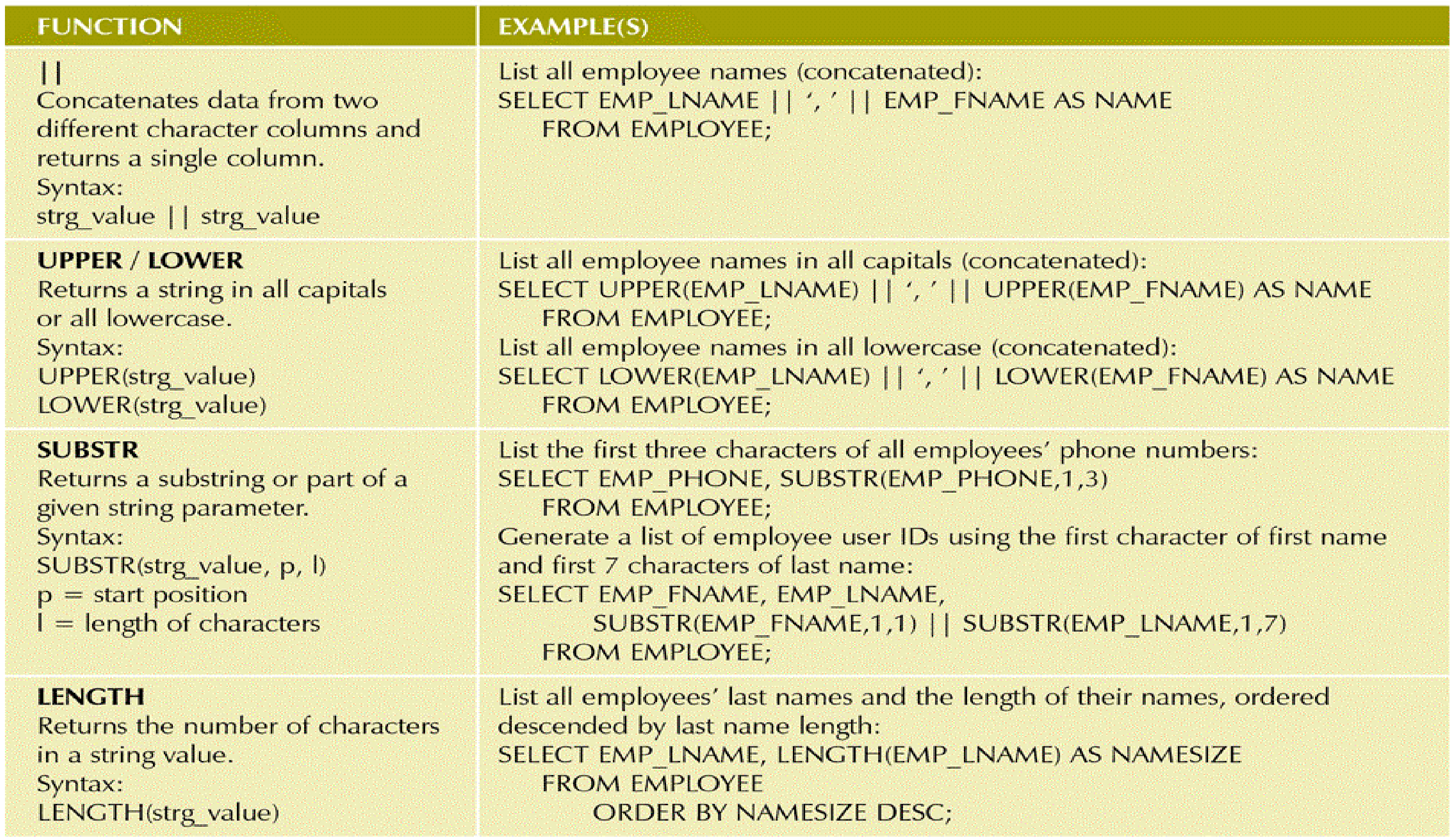


TABLE 7.5 SELECTED ORACLE NUMERIC FUNCTIONS

FUNCTION	EXAMPLE(S)
ABS Returns the absolute value of a number. Syntax: ABS(numeric_value)	List 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	List 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	List 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)	List 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;

Selected Oracle String Functions

TABLE 7.6 SELECTED ORACLE STRING FUNCTIONS



FUNCTION	EXAMPLE(S)

||
Concatenates data from two different character columns and returns a single column.
Syntax:
strg_value || strg_value
	List all employee names (concatenated):
	SELECT EMP_LNAME || ', ' || EMP_FNAME AS NAME
	FROM EMPLOYEE;

UPPER / LOWER
Returns a string in all capitals or all lowercase.
Syntax:
UPPER(strg_value)
LOWER(strg_value)
	List all employee names in all capitals (concatenated):
	SELECT UPPER(EMP_LNAME) || ', ' || UPPER(EMP_FNAME) AS NAME
	FROM EMPLOYEE;
	List all employee names in all lowercase (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
	List the first three characters of all employees' phone numbers:
	SELECT EMP_PHONE, SUBSTR(EMP_PHONE,1,3)
	FROM EMPLOYEE;
	Generate a list of employee user IDs using the first character of first name and first 7 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)
	List all employees' 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;

Selected Oracle Conversion Functions

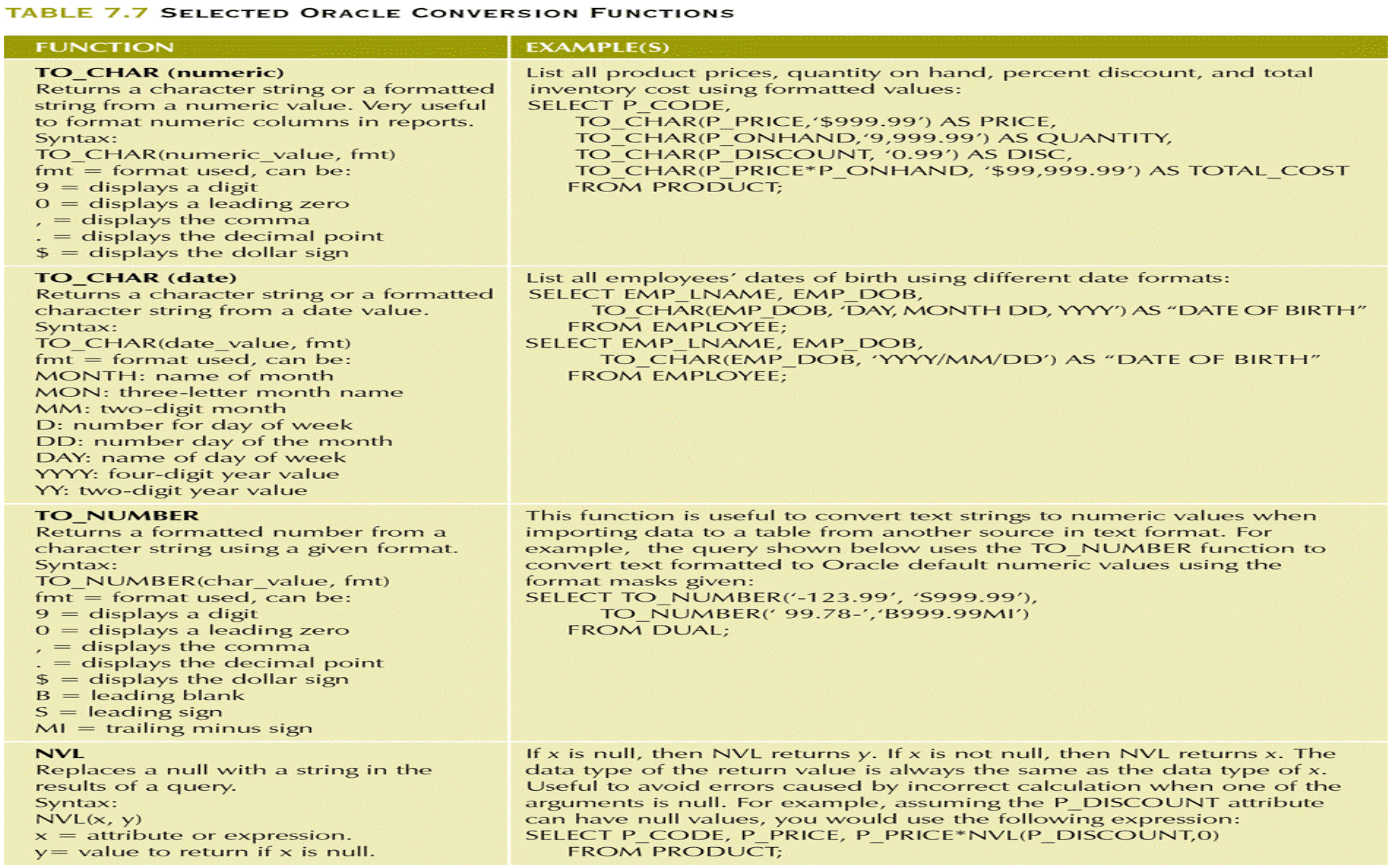


TABLE 7.7 SELECTED ORACLE CONVERSION FUNCTIONS

FUNCTION	EXAMPLE(S)
TO_CHAR (numeric) Returns a character string or a formatted string from a numeric value. Very useful to format 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	List 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_ONHAND,'9,999.99') AS QUANTITY, TO_CHAR(P_DISCOUNT, '0.99') AS DISC, TO_CHAR(P_PRICE*P_ONHAND, '$99,999.99') AS TOTAL_COST FROM PRODUCT;

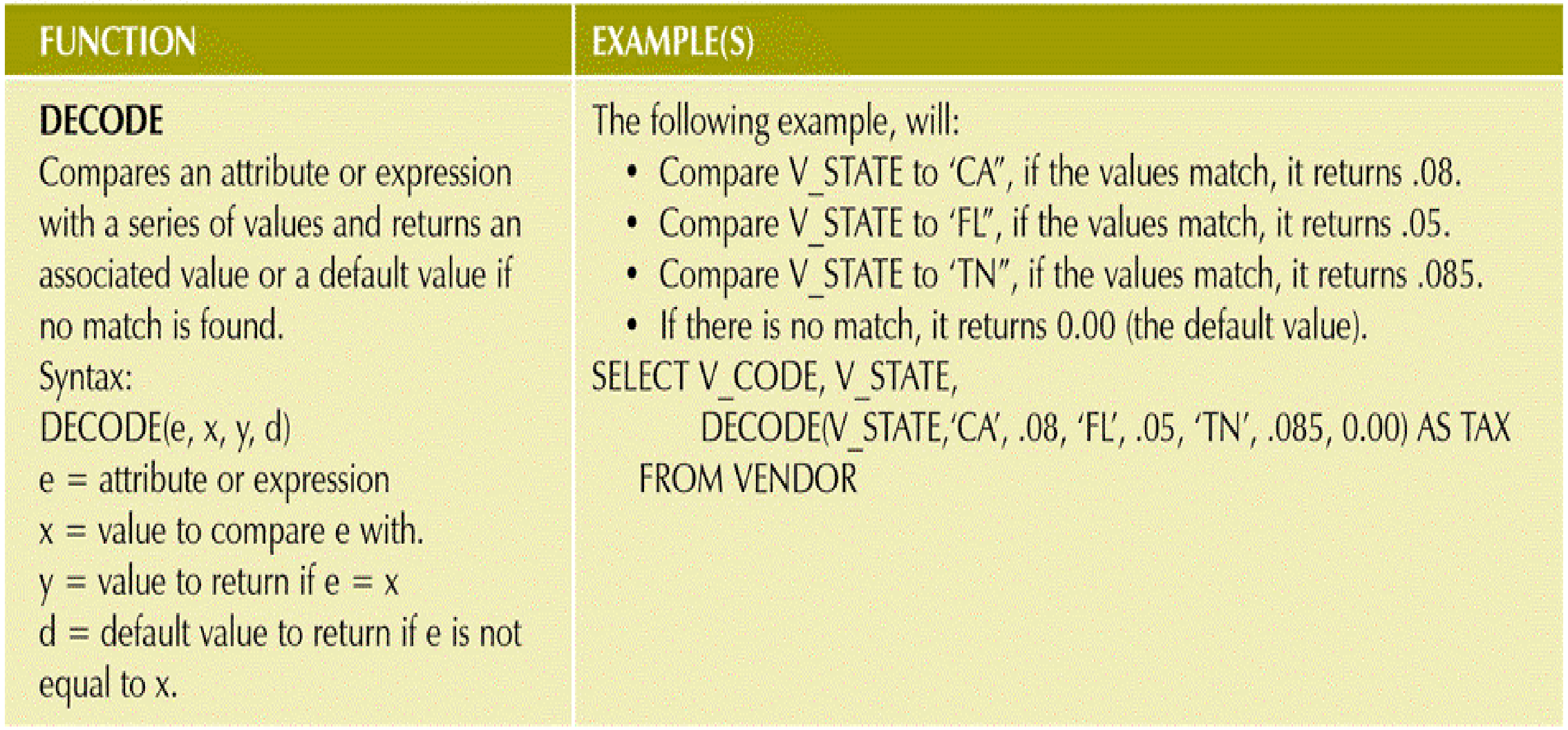
TO_CHAR (date)
Returns a character string or a formatted character string from a date value.
Syntax:
TO_CHAR(date_value, fmt)
fmt = format used, can be:
MONTH: name of month
MON: three-letter month name
MM: two-digit month
D: number for day of week
DD: number day of the month
DAY: name of day of week
YYYY: four-digit year value
YY: two-digit year value
	List all employees' 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
	This function is useful to convert 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 to avoid 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;

Selected Oracle Conversion Functions (continued)

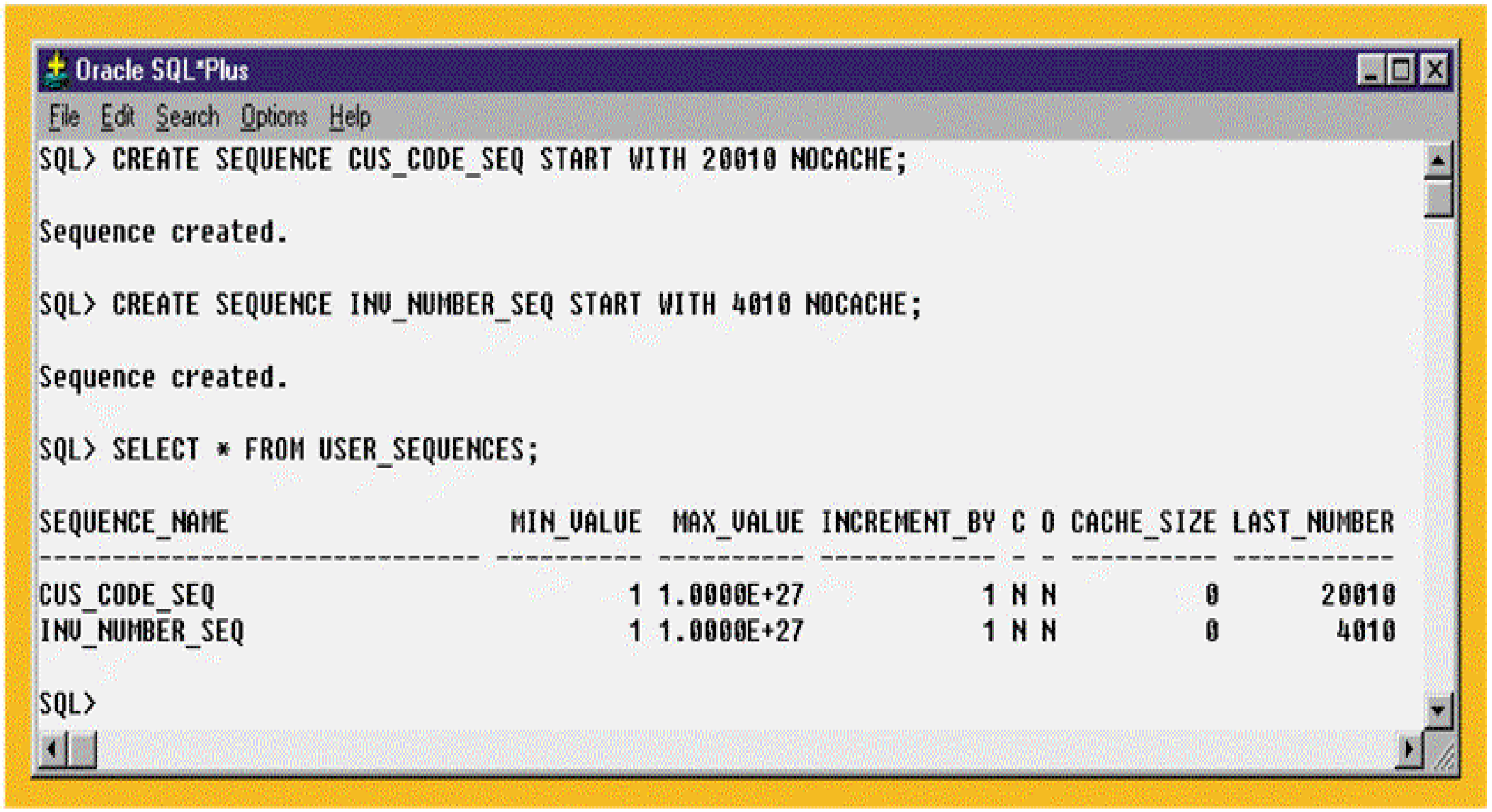
TABLE 7.7 SELECTED ORACLE CONVERSION FUNCTIONS (CONTINUED)



FUNCTION	EXAMPLE(S)
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 to compare e with. y = value to return if e = x d = default value to return if e is not equal to x.	The following example, will: • Compare V_STATE to 'CA", if the values match, it returns .08. • Compare V_STATE to 'FL", if the values match, it returns .05. • Compare 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

Oracle Sequence

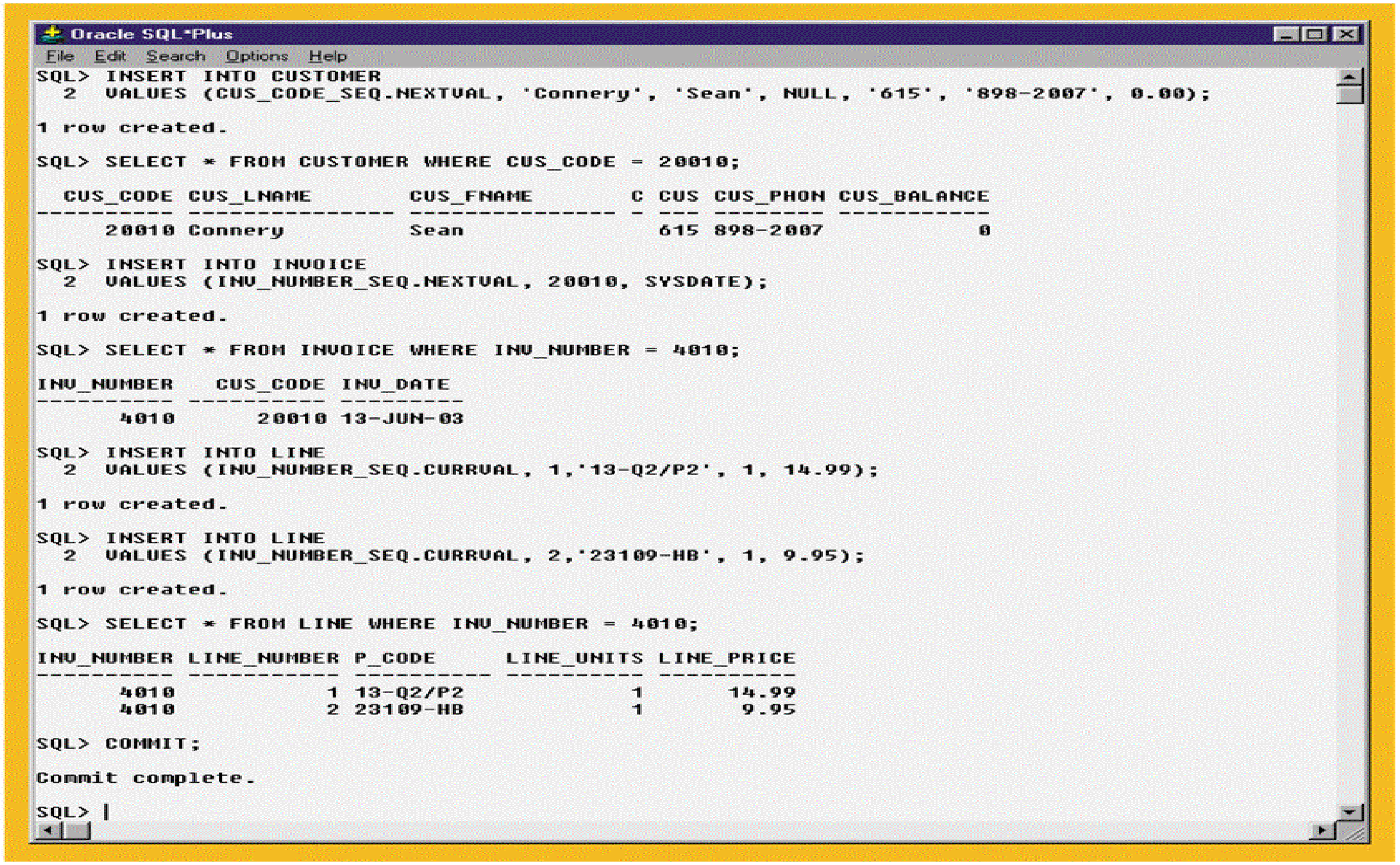
FIGURE 7.22 ORACLE SEQUENCE



```
SQL> CREATE SEQUENCE CUS_CODE_SEQ START WITH 20010 NOCACHE;

Sequence created.

SQL> CREATE SEQUENCE INV_NUMBER_SEQ START WITH 4010 NOCACHE;

Sequence created.

SQL> SELECT * FROM USER_SEQUENCES;

SEQUENCE_NAME                   MIN_VALUE  MAX_VALUE INCREMENT_BY C O CACHE_SIZE LAST_NUMBER
------------------------------ ---------- ---------- ------------ - - ---------- -----------
CUS_CODE_SEQ                            1 1.0000E+27            1 N N          0       20010
INV_NUMBER_SEQ                          1 1.0000E+27            1 N N          0        4010

SQL>
```

Oracle Sequence Examples

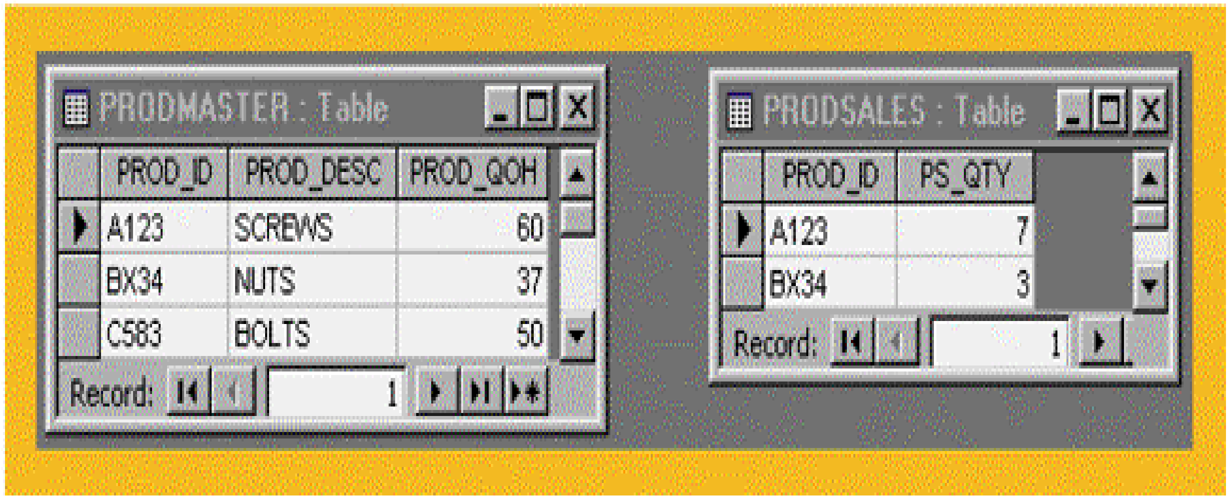
FIGURE 7.23 ORACLE SEQUENCE EXAMPLES



```
SQL> INSERT INTO CUSTOMER
  2  VALUES (CUS_CODE_SEQ.NEXTVAL, 'Connery', 'Sean', NULL, '615', '898-2007', 0.00);

1 row created.

SQL> SELECT * FROM CUSTOMER WHERE CUS_CODE = 20010;

  CUS_CODE CUS_LNAME       CUS_FNAME       C CUS CUS_PHON CUS_BALANCE
---------- --------------- --------------- - --- -------- -----------
     20010 Connery         Sean              615 898-2007           0

SQL> INSERT INTO INVOICE
  2  VALUES (INV_NUMBER_SEQ.NEXTVAL, 20010, SYSDATE);

1 row created.

SQL> SELECT * FROM INVOICE WHERE INV_NUMBER = 4010;

INV_NUMBER   CUS_CODE INV_DATE
---------- ---------- ---------
      4010      20010 13-JUN-03

SQL> INSERT INTO LINE
  2  VALUES (INV_NUMBER_SEQ.CURRVAL, 1,'13-Q2/P2', 1, 14.99);

1 row created.

SQL> INSERT INTO LINE
  2  VALUES (INV_NUMBER_SEQ.CURRVAL, 2,'23109-HB', 1, 9.95);

1 row created.

SQL> SELECT * FROM LINE WHERE INV_NUMBER = 4010;

INV_NUMBER LINE_NUMBER P_CODE     LINE_UNITS LINE_PRICE
---------- ----------- ---------- ---------- ----------
      4010           1 13-Q2/P2            1      14.99
      4010           2 23109-HB            1       9.95

SQL> COMMIT;

Commit complete.

SQL> |
```

The PRODMASTER and PRODSALES Tables

FIGURE 7.24 THE PRODMASTER AND PRODSALES TABLES



PRODMASTER : Table

PROD_ID	PROD_DESC	PROD_QOH
A123	SCREWS	60
BX34	NUTS	37
C583	BOLTS	50

Record: 1

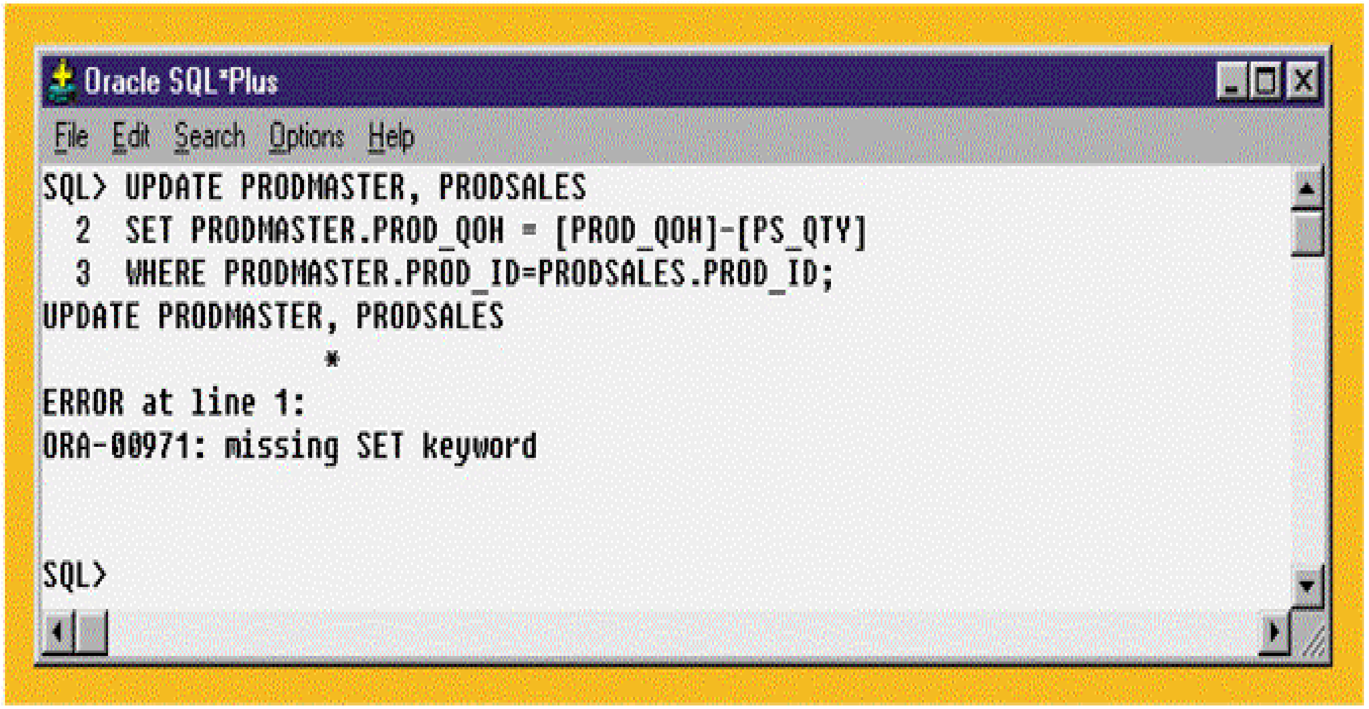
PRODSALES : Table

PROD_ID	PS_QTY
A123	7
BX34	3

Record: 1

The Oracle UPDATE Error Message

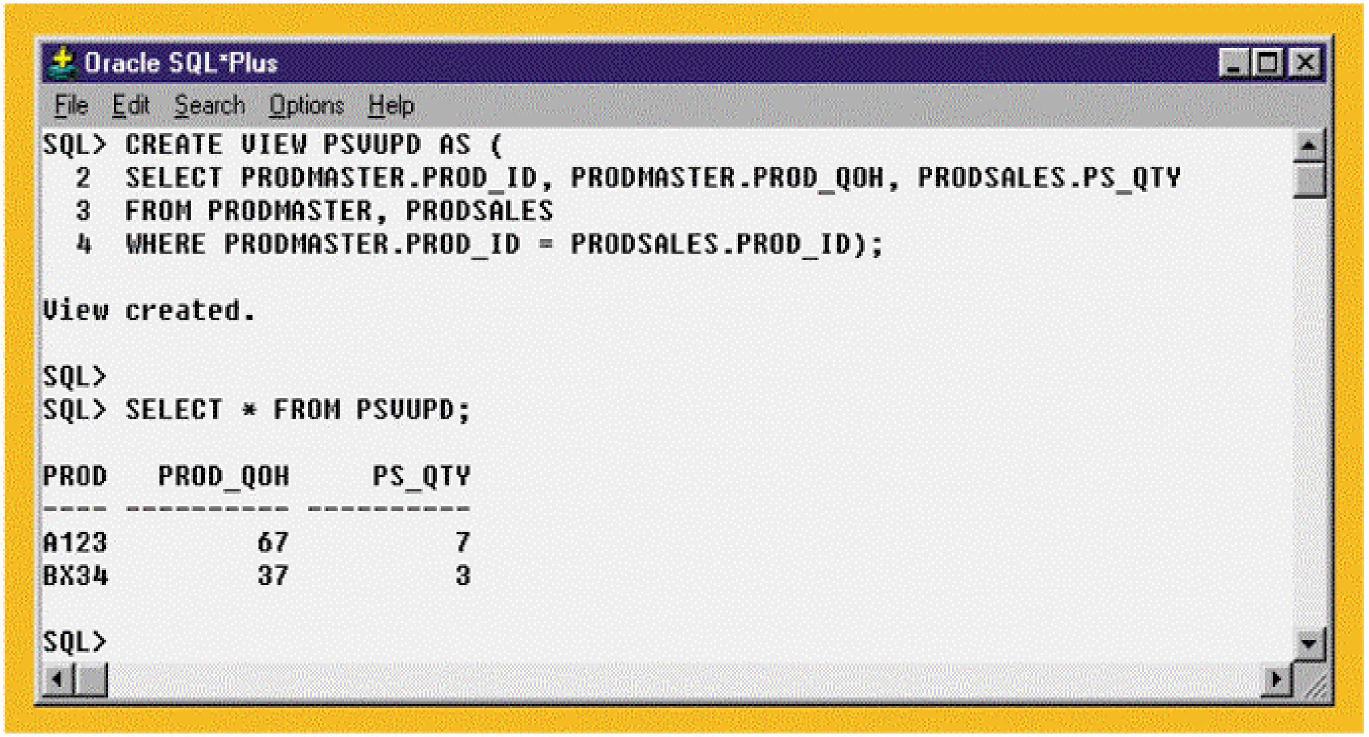
FIGURE 7.25 THE ORACLE UPDATE ERROR MESSAGE



```
SQL> UPDATE PRODMASTER, PRODSALES
  2  SET PRODMASTER.PROD_QOH = [PROD_QOH]-[PS_QTY]
  3  WHERE PRODMASTER.PROD_ID=PRODSALES.PROD_ID;
UPDATE PRODMASTER, PRODSALES
                 *
ERROR at line 1:
ORA-00971: missing SET keyword


SQL>
```

Creating an Updatable View in Oracle

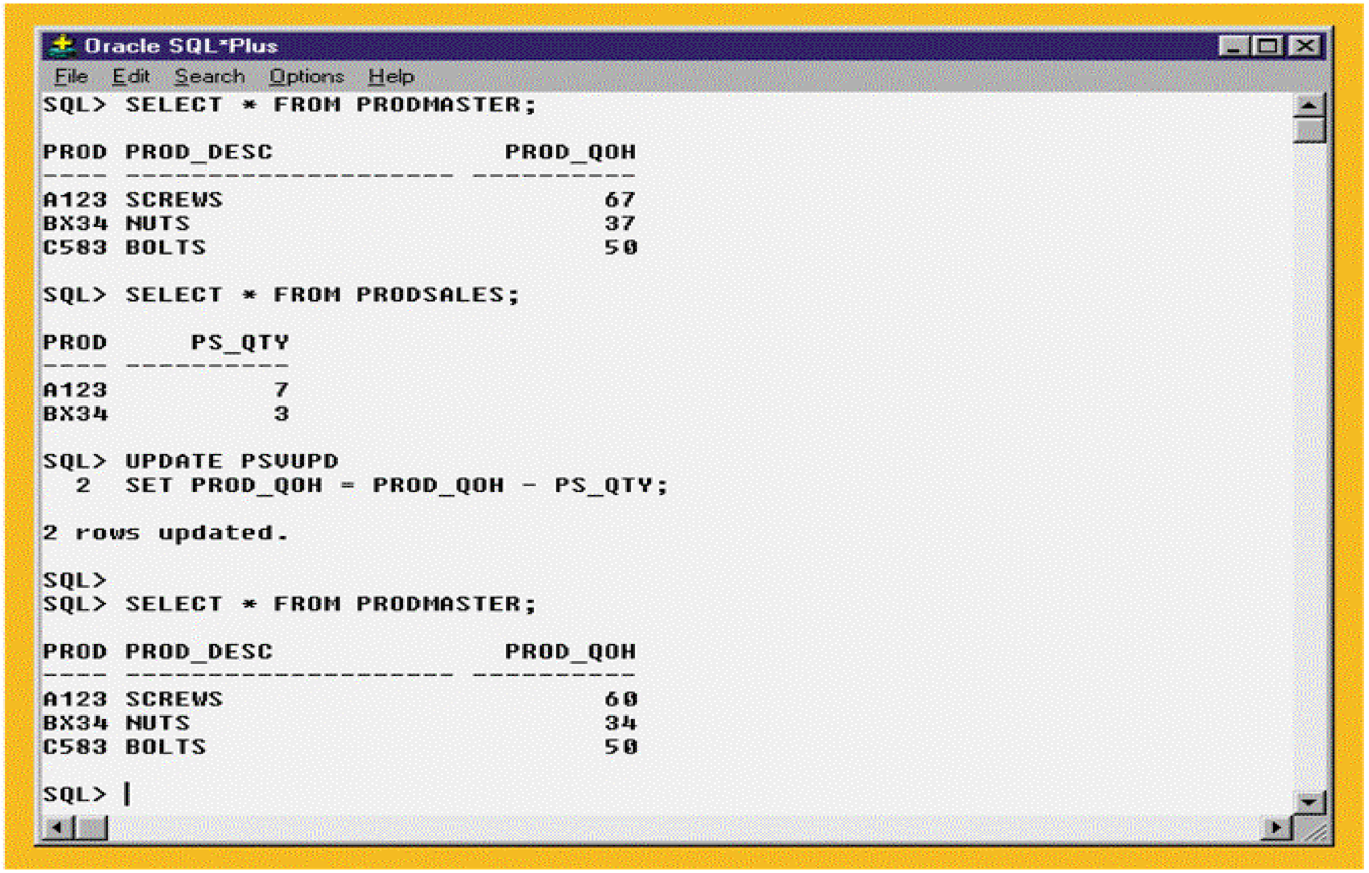
FIGURE 7.26 CREATING AN UPDATABLE VIEW IN ORACLE



```
SQL> CREATE VIEW PSVUPD AS (
  2  SELECT PRODMASTER.PROD_ID, PRODMASTER.PROD_QOH, PRODSALES.PS_QTY
  3  FROM PRODMASTER, PRODSALES
  4  WHERE PRODMASTER.PROD_ID = PRODSALES.PROD_ID);

View created.

SQL>
SQL> SELECT * FROM PSVUPD;

PROD   PROD_QOH     PS_QTY
---- ---------- ----------
A123         67          7
BX34         37          3

SQL>
```

PRODMASTER Table Update, Using an Updatable View

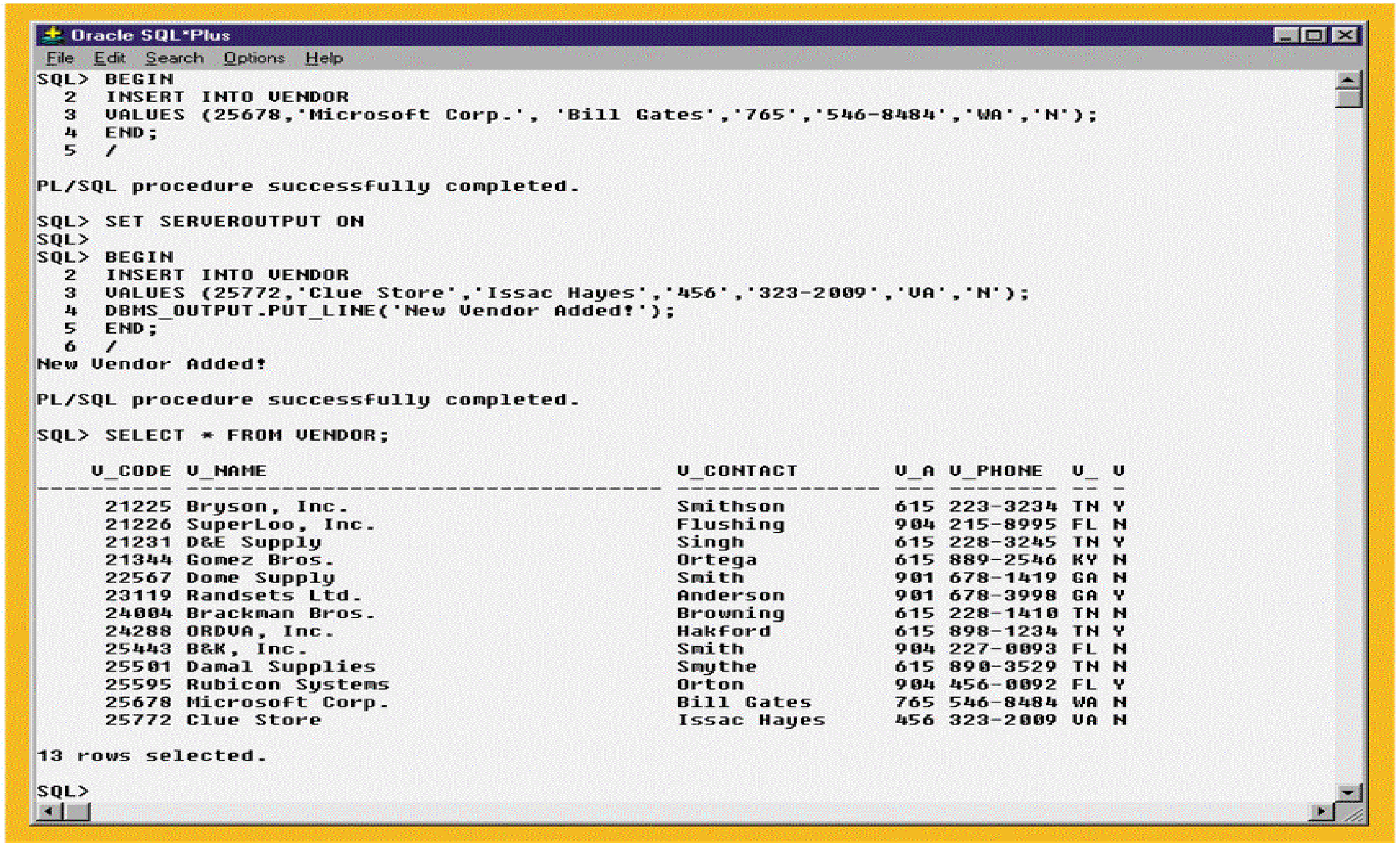
FIGURE 7.27 PRODMASTER TABLE UPDATE, USING AN UPDATABLE VIEW



```
SQL> SELECT * FROM PRODMASTER;

PROD PROD_DESC                     PROD_QOH
---- -------------------- ----------
A123 SCREWS                              67
BX34 NUTS                                37
C583 BOLTS                               50

SQL> SELECT * FROM PRODSALES;

PROD     PS_QTY
---- ----------
A123          7
BX34          3

SQL> UPDATE PSVUPD
  2  SET PROD_QOH = PROD_QOH - PS_QTY;

2 rows updated.

SQL>
SQL> SELECT * FROM PRODMASTER;

PROD PROD_DESC                     PROD_QOH
---- -------------------- ----------
A123 SCREWS                              60
BX34 NUTS                                34
C583 BOLTS                               50

SQL> |
```

Anonymous PL/SQL Block Examples

FIGURE 7.28 ANONYMOUS PL/SQL BLOCK EXAMPLES



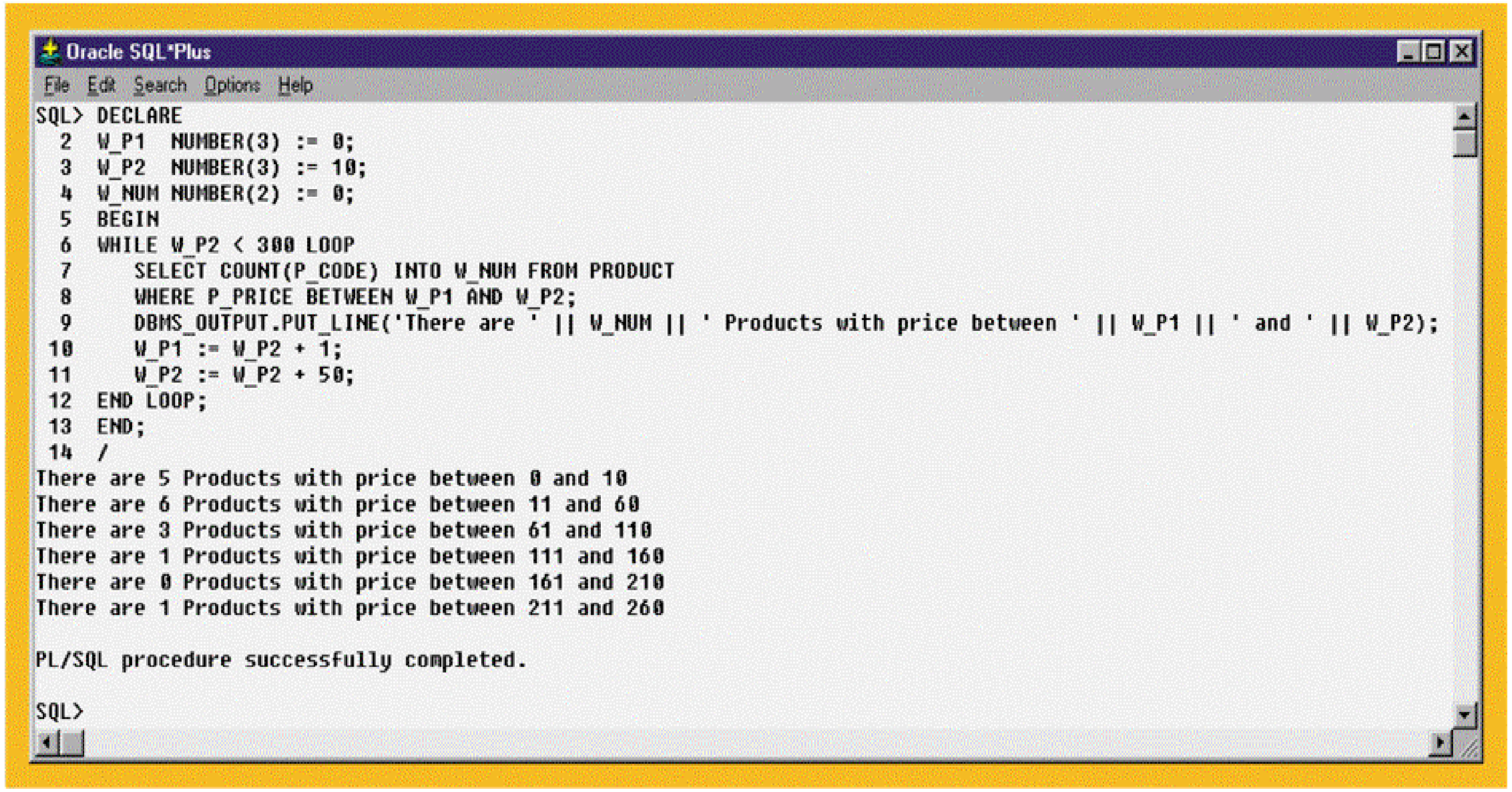
```
SQL> BEGIN
  2  INSERT INTO VENDOR
  3  VALUES (25678,'Microsoft Corp.', 'Bill Gates','765','546-8484','WA','N');
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL> SET SERVEROUTPUT ON
SQL>
SQL> BEGIN
  2  INSERT INTO VENDOR
  3  VALUES (25772,'Clue Store','Issac Hayes','456','323-2009','VA','N');
  4  DBMS_OUTPUT.PUT_LINE('New Vendor Added!');
  5  END;
  6  /
New Vendor Added!

PL/SQL procedure successfully completed.

SQL> SELECT * FROM VENDOR;

    V_CODE V_NAME                              V_CONTACT       V_A V_PHONE  V_ V
---------- ----------------------------------- --------------- --- -------- -- -
     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
     25678 Microsoft Corp.                     Bill Gates      765 546-8484 WA N
     25772 Clue Store                          Issac Hayes     456 323-2009 VA N

13 rows selected.

SQL>
```

SHOW ERRORS

- Can help diagnose errors found in PL/SQL blocks
- Yields additional debugging information whenever an error is generated after an PL/SQL block is created or executed

Anonymous PL/SQL Block with Variables and Loops

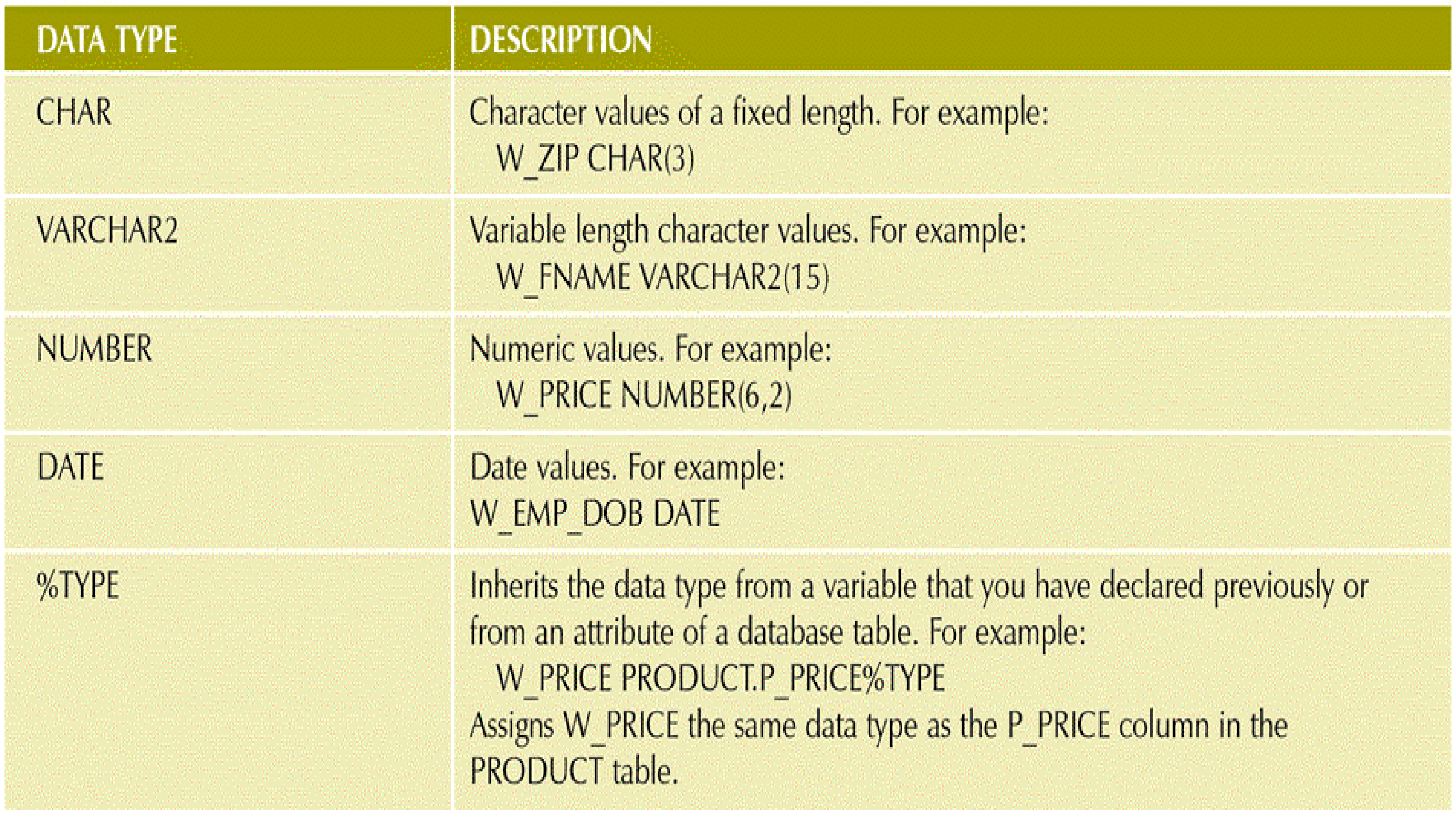
FIGURE 7.29 ANONYMOUS PL/SQL BLOCK WITH VARIABLES AND LOOPS



```
SQL> DECLARE
  2  W_P1   NUMBER(3) := 0;
  3  W_P2   NUMBER(3) := 10;
  4  W_NUM NUMBER(2) := 0;
  5  BEGIN
  6  WHILE W_P2 < 300 LOOP
  7     SELECT COUNT(P_CODE) INTO W_NUM FROM PRODUCT
  8     WHERE P_PRICE BETWEEN W_P1 AND W_P2;
  9     DBMS_OUTPUT.PUT_LINE('There are ' || W_NUM || ' Products with price between ' || W_P1 || ' and ' || W_P2);
 10     W_P1 := W_P2 + 1;
 11     W_P2 := W_P2 + 50;
 12  END LOOP;
 13  END;
 14  /
There are 5 Products with price between 0 and 10
There are 6 Products with price between 11 and 60
There are 3 Products with price between 61 and 110
There are 1 Products with price between 111 and 160
There are 0 Products with price between 161 and 210
There are 1 Products with price between 211 and 260

PL/SQL procedure successfully completed.

SQL>
```

PL/SQL Basic Data Types

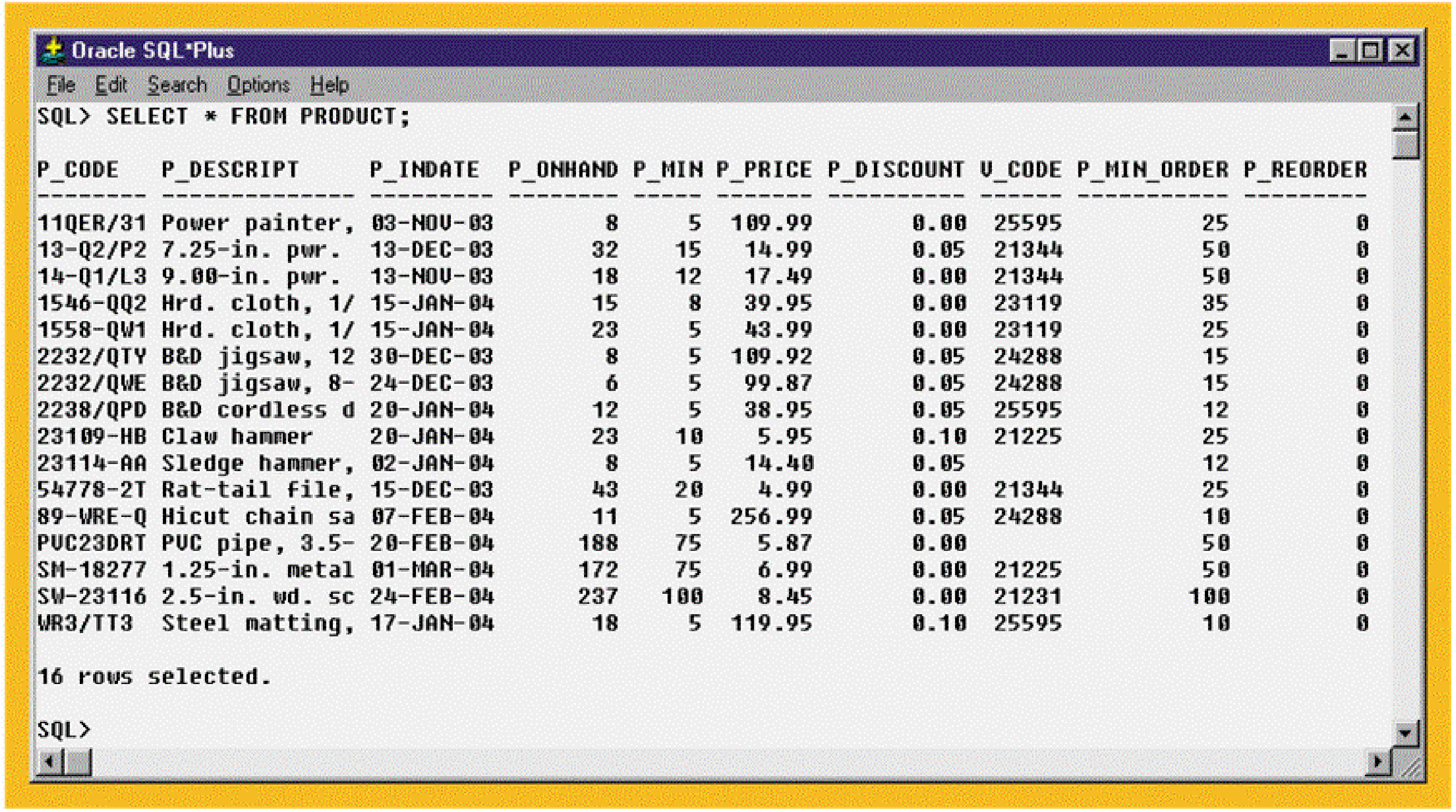
TABLE 7.8 PL/SQL BASIC DATA TYPES



DATA TYPE	DESCRIPTION
CHAR	Character values of a fixed length. For example: W_ZIP CHAR(3)
VARCHAR2	Variable length character values. For example: W_FNAME VARCHAR2(15)
NUMBER	Numeric values. For example: W_PRICE NUMBER(6,2)
DATE	Date values. For example: W_EMP_DOB DATE
%TYPE	Inherits the data type from a variable that you have declared previously or from an attribute of a database table. For example: W_PRICE PRODUCT.P_PRICE%TYPE Assigns W_PRICE the same data type as the P_PRICE column in the PRODUCT table.

The PRODUCT Table

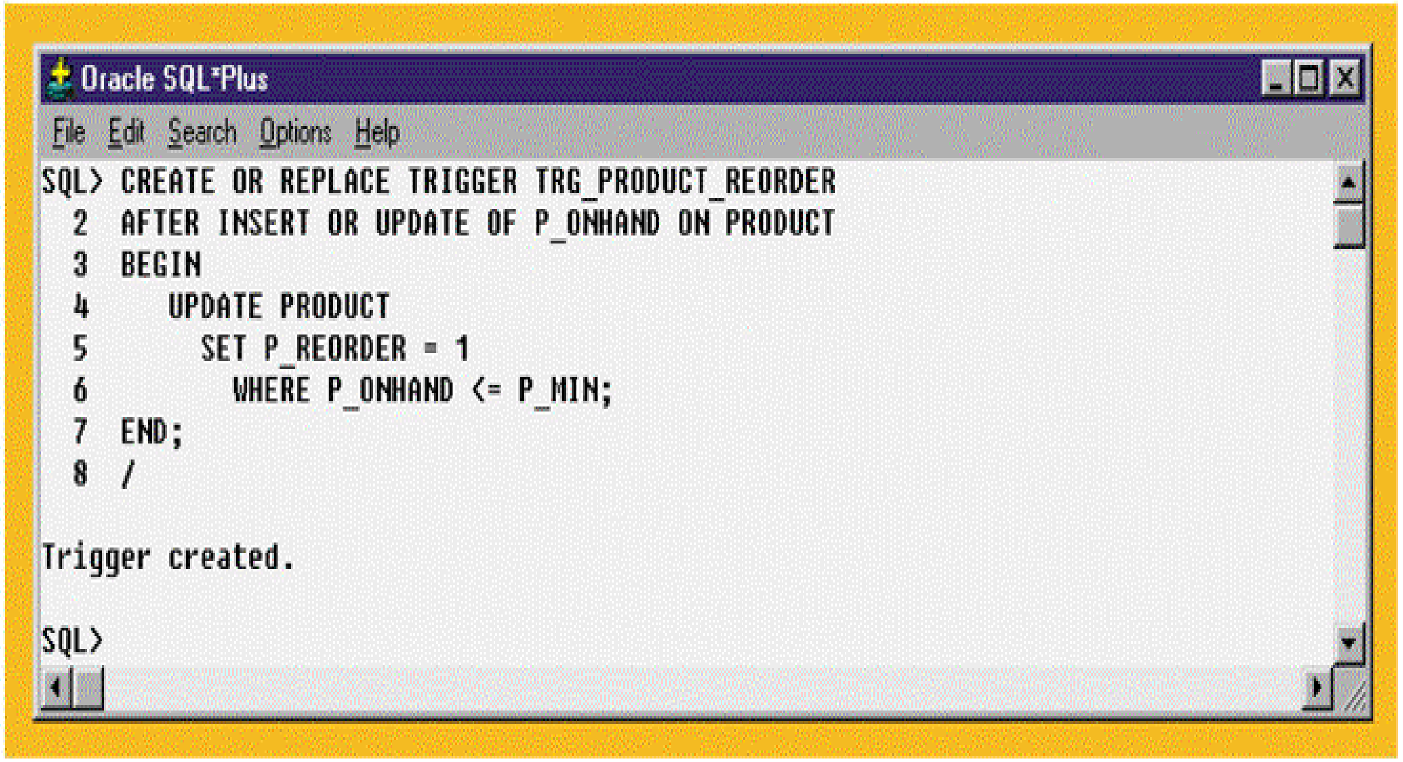
FIGURE 7.30 THE PRODUCT TABLE



```
SQL> SELECT * FROM PRODUCT;

P_CODE   P_DESCRIPT     P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT V_CODE P_MIN_ORDER P_REORDER
-------- -------------- --------- -------- ----- ------- ---------- ------ ----------- ---------
11QER/31 Power painter, 03-NOV-03        8     5  109.99       0.00  25595          25         0
13-Q2/P2 7.25-in. pwr.  13-DEC-03       32    15   14.99       0.05  21344          50         0
14-Q1/L3 9.00-in. pwr.  13-NOV-03       18    12   17.49       0.00  21344          50         0
1546-QQ2 Hrd. cloth, 1/ 15-JAN-04       15     8   39.95       0.00  23119          35         0
1558-QW1 Hrd. cloth, 1/ 15-JAN-04       23     5   43.99       0.00  23119          25         0
2232/QTY B&D jigsaw, 12 30-DEC-03        8     5  109.92       0.05  24288          15         0
2232/QWE B&D jigsaw, 8- 24-DEC-03        6     5   99.87       0.05  24288          15         0
2238/QPD B&D cordless d 20-JAN-04       12     5   38.95       0.05  25595          12         0
23109-HB Claw hammer    20-JAN-04       23    10    5.95       0.10  21225          25         0
23114-AA Sledge hammer, 02-JAN-04        8     5   14.40       0.05                 12         0
54778-2T Rat-tail file, 15-DEC-03       43    20    4.99       0.00  21344          25         0
89-WRE-Q Hicut chain sa 07-FEB-04       11     5  256.99       0.05  24288          10         0
PVC23DRT PVC pipe, 3.5- 20-FEB-04      188    75    5.87       0.00                 50         0
SM-18277 1.25-in. metal 01-MAR-04      172    75    6.99       0.00  21225          50         0
SW-23116 2.5-in. wd. sc 24-FEB-04      237   100    8.45       0.00  21231         100         0
WR3/TT3  Steel matting, 17-JAN-04       18     5  119.95       0.10  25595          10         0

16 rows selected.

SQL>
```

Creating the TRG_PRODUCT_REORDER Trigger

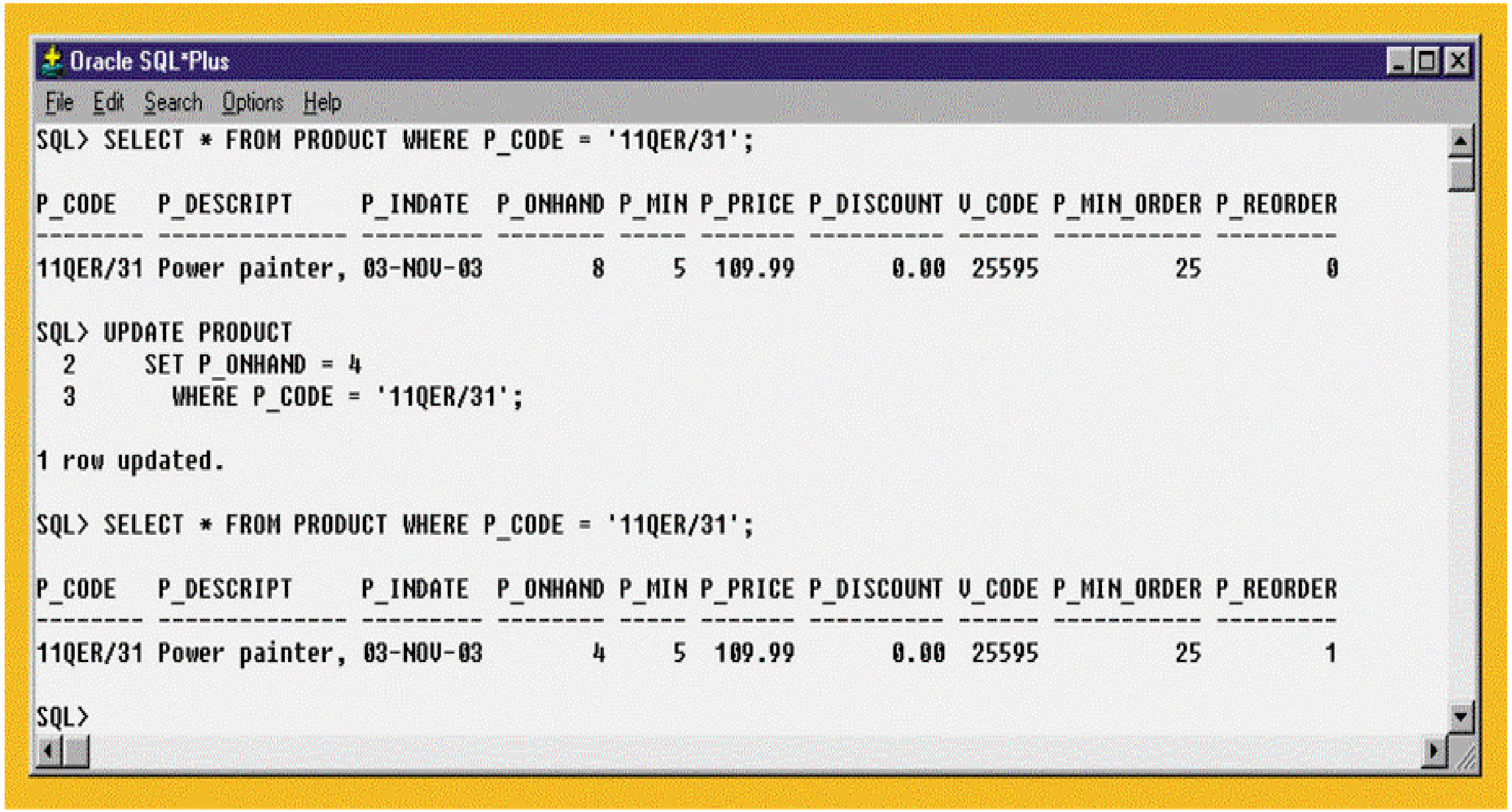
FIGURE 7.31 CREATING THE TRG_PRODUCT_REORDER TRIGGER



```
SQL> CREATE OR REPLACE TRIGGER TRG_PRODUCT_REORDER
  AFTER INSERT OR UPDATE OF P_ONHAND ON PRODUCT
  BEGIN
     UPDATE PRODUCT
       SET P_REORDER = 1
         WHERE P_ONHAND <= P_MIN;
  END;
  /

Trigger created.

SQL>
```

Verifying the TRG_PRODUCT_REORDER Trigger Execution

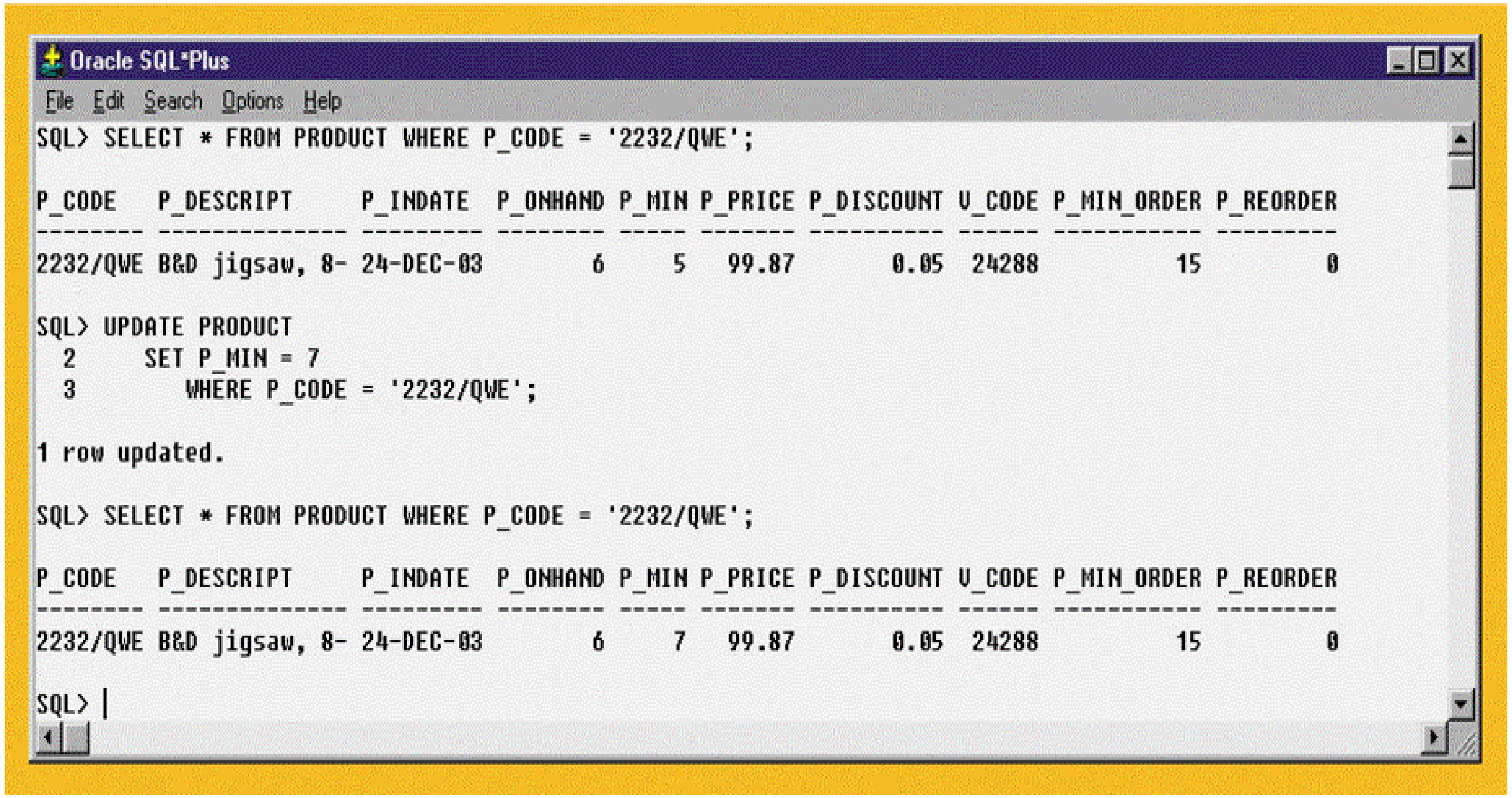
FIGURE 7.32 VERIFYING THE TRG_PRODUCT_REORDER TRIGGER EXECUTION



```
SQL> SELECT * FROM PRODUCT WHERE P_CODE = '11QER/31';

P_CODE   P_DESCRIPT     P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT V_CODE P_MIN_ORDER P_REORDER
-------- -------------- --------- -------- ----- ------- ---------- ------ ----------- ---------
11QER/31 Power painter, 03-NOV-03        8     5  109.99       0.00  25595          25         0

SQL> UPDATE PRODUCT
  2     SET P_ONHAND = 4
  3       WHERE P_CODE = '11QER/31';

1 row updated.

SQL> SELECT * FROM PRODUCT WHERE P_CODE = '11QER/31';

P_CODE   P_DESCRIPT     P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT V_CODE P_MIN_ORDER P_REORDER
-------- -------------- --------- -------- ----- ------- ---------- ------ ----------- ---------
11QER/31 Power painter, 03-NOV-03        4     5  109.99       0.00  25595          25         1

SQL>
```

The P_REORDER Value Mismatch After Update of the P_MIN Attribute

FIGURE 7.33 THE P_REORDER VALUE MISMATCH AFTER UPDATE OF THE P_MIN ATTRIBUTE



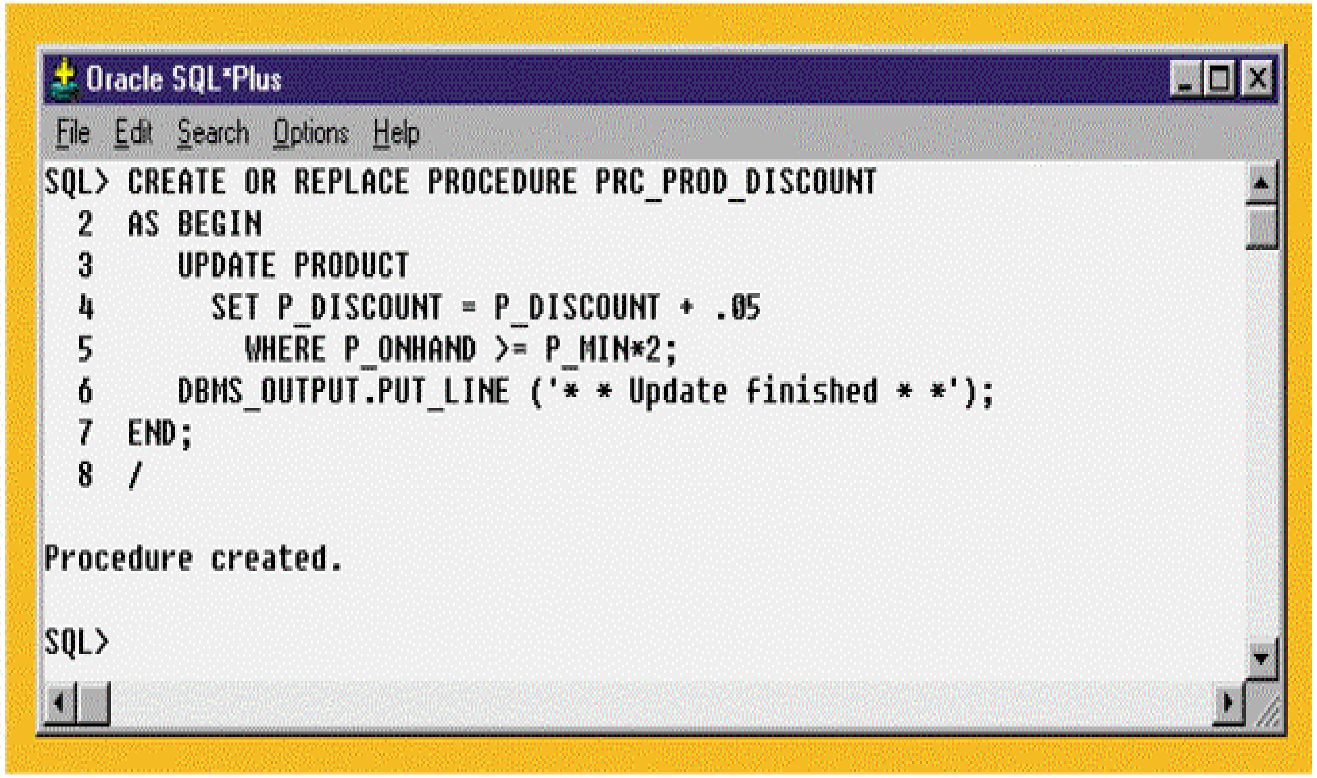
```
SQL> SELECT * FROM PRODUCT WHERE P_CODE = '2232/QWE';

P_CODE   P_DESCRIPT     P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT V_CODE P_MIN_ORDER P_REORDER
-------- -------------- --------- -------- ----- ------- ---------- ------ ----------- ---------
2232/QWE B&D jigsaw, 8- 24-DEC-03        6     5   99.87       0.05  24288          15         0

SQL> UPDATE PRODUCT
  2     SET P_MIN = 7
  3        WHERE P_CODE = '2232/QWE';

1 row updated.

SQL> SELECT * FROM PRODUCT WHERE P_CODE = '2232/QWE';

P_CODE   P_DESCRIPT     P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT V_CODE P_MIN_ORDER P_REORDER
-------- -------------- --------- -------- ----- ------- ---------- ------ ----------- ---------
2232/QWE B&D jigsaw, 8- 24-DEC-03        6     7   99.87       0.05  24288          15         0

SQL> |
```

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

Creating the PRC_PROD_DISCOUNT Stored Procedure

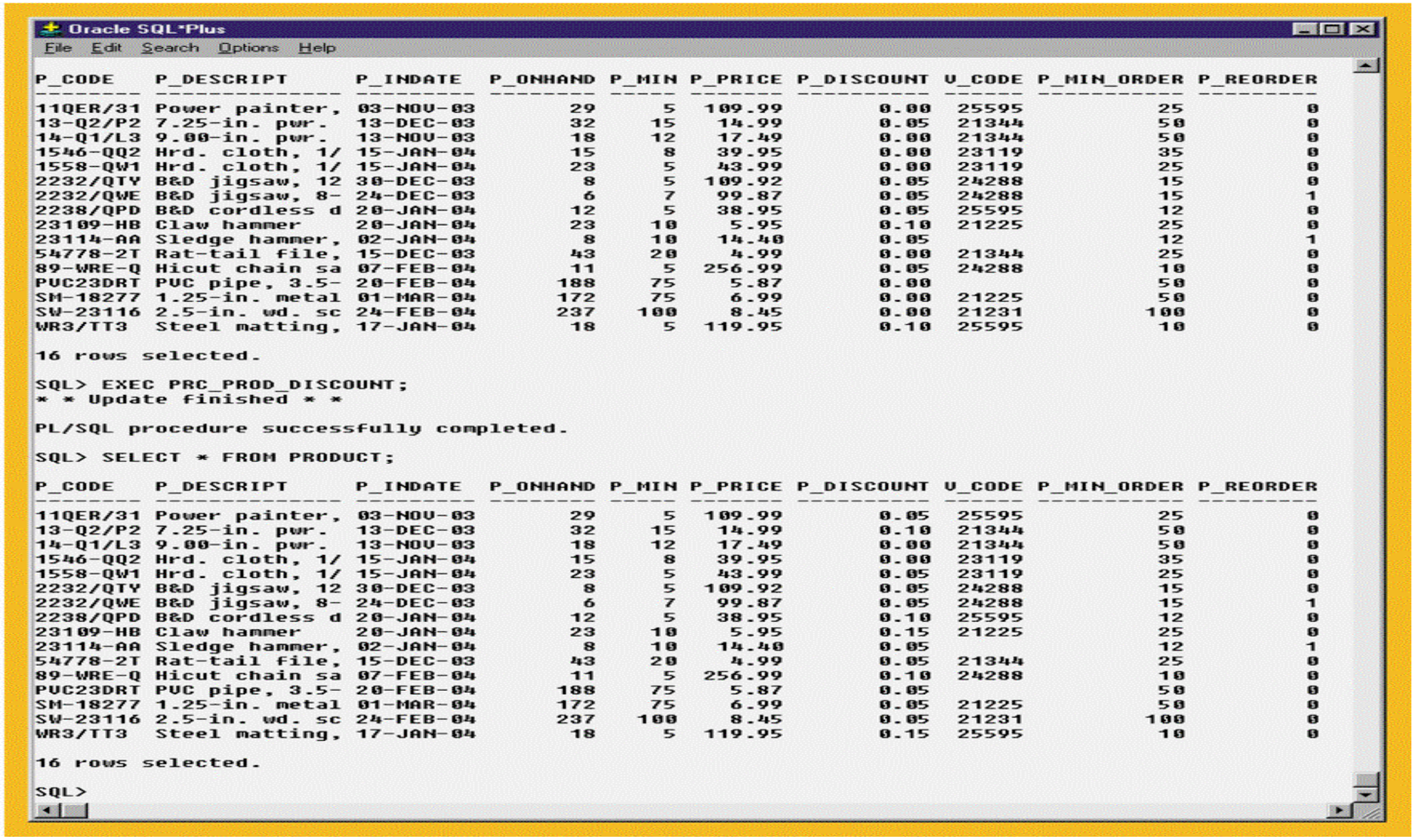
FIGURE 7.41 CREATING THE PRC_PROD_DISCOUNT STORED PROCEDURE



```
SQL> CREATE OR REPLACE PROCEDURE PRC_PROD_DISCOUNT
  AS BEGIN
     UPDATE PRODUCT
       SET P_DISCOUNT = P_DISCOUNT + .05
         WHERE P_ONHAND >= P_MIN*2;
     DBMS_OUTPUT.PUT_LINE ('* * Update finished * *');
  END;
  /

Procedure created.

SQL>
```

Results of the PRC_PROD_DISCOUNT Stored Procedure

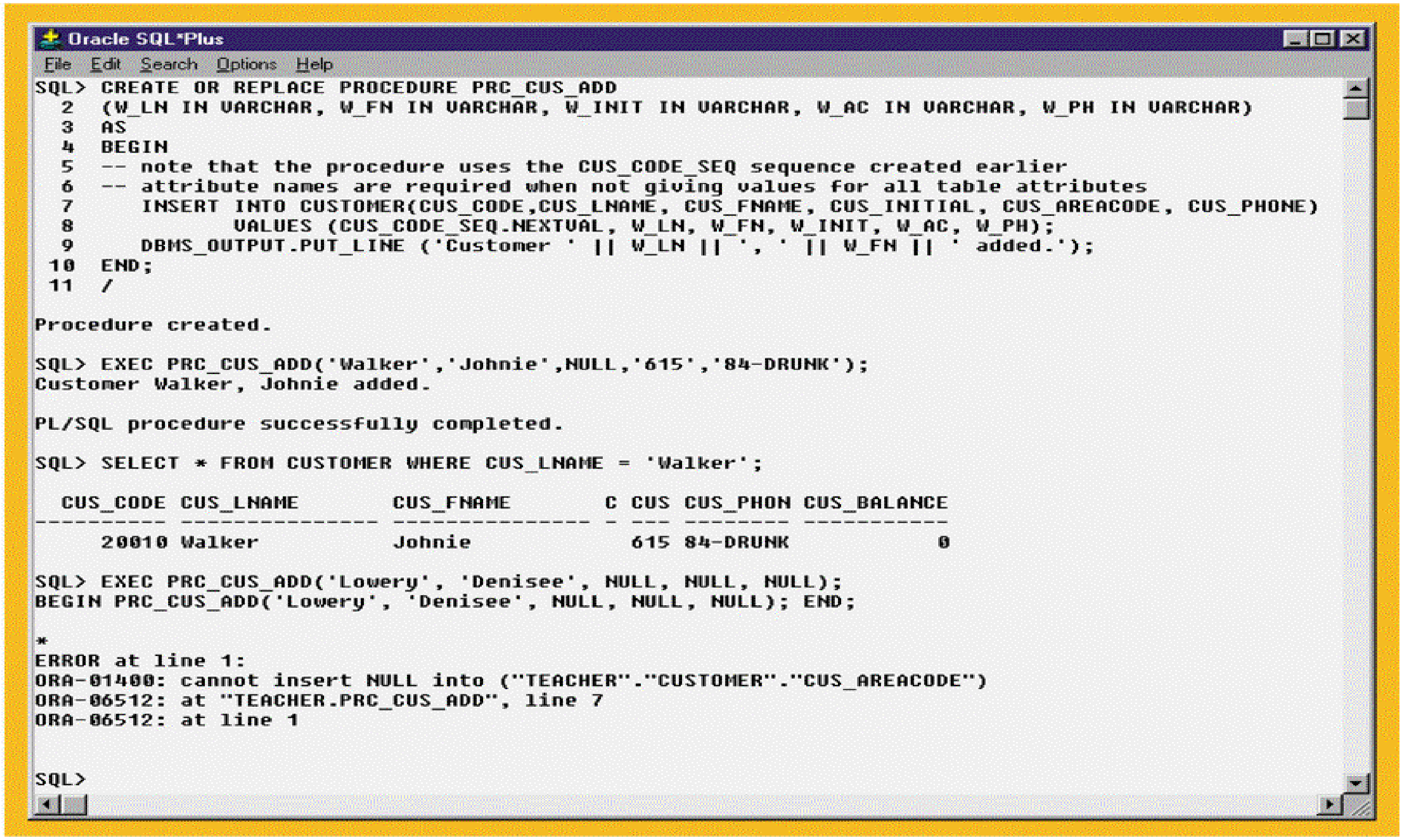
FIGURE 7.42 Results of the PRC_PROD_DISCOUNT Stored Procedure



```
P_CODE     P_DESCRIPT      P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT V_CODE P_MIN_ORDER P_REORDER
---------- --------------- --------- -------- ----- ------- ---------- ------ ----------- ---------
11QER/31   Power painter,  03-NOV-03       29     5  109.99       0.00  25595          25         0
13-Q2/P2   7.25-in. pwr.   13-DEC-03       32    15   14.99       0.05  21344          50         0
14-Q1/L3   9.00-in. pwr.   13-NOV-03       18    12   17.49       0.00  21344          50         0
1546-QQ2   Hrd. cloth, 1/  15-JAN-04       15     8   39.95       0.00  23119          35         0
1558-QW1   Hrd. cloth, 1/  15-JAN-04       23     5   43.99       0.00  23119          25         0
2232/QTY   B&D jigsaw, 12  30-DEC-03        8     5  109.92       0.05  24288          15         0
2232/QWE   B&D jigsaw, 8-  24-DEC-03        6     7   99.87       0.05  24288          15         1
2238/QPD   B&D cordless d  20-JAN-04       12     5   38.95       0.05  25595          12         0
23109-HB   Claw hammer     20-JAN-04       23    10    5.95       0.10  21225          25         0
23114-AA   Sledge hammer,  02-JAN-04        8    10   14.40       0.05                 12         1
54778-2T   Rat-tail file,  15-DEC-03       43    20    4.99       0.00  21344          25         0
89-WRE-Q   Hicut chain sa  07-FEB-04       11     5  256.99       0.05  24288          10         0
PVC23DRT   PVC pipe, 3.5-  20-FEB-04      188    75    5.87       0.00                 50         0
SM-18277   1.25-in. metal  01-MAR-04      172    75    6.99       0.00  21225          50         0
SW-23116   2.5-in. wd. sc  24-FEB-04      237   100    8.45       0.00  21231         100         0
WR3/TT3    Steel matting,  17-JAN-04       18     5  119.95       0.10  25595          10         0

16 rows selected.

SQL> EXEC PRC_PROD_DISCOUNT;
* * Update finished * *

PL/SQL procedure successfully completed.

SQL> SELECT * FROM PRODUCT;

P_CODE     P_DESCRIPT      P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT V_CODE P_MIN_ORDER P_REORDER
---------- --------------- --------- -------- ----- ------- ---------- ------ ----------- ---------
11QER/31   Power painter,  03-NOV-03       29     5  109.99       0.05  25595          25         0
13-Q2/P2   7.25-in. pwr.   13-DEC-03       32    15   14.99       0.10  21344          50         0
14-Q1/L3   9.00-in. pwr.   13-NOV-03       18    12   17.49       0.00  21344          50         0
1546-QQ2   Hrd. cloth, 1/  15-JAN-04       15     8   39.95       0.00  23119          35         0
1558-QW1   Hrd. cloth, 1/  15-JAN-04       23     5   43.99       0.05  23119          25         0
2232/QTY   B&D jigsaw, 12  30-DEC-03        8     5  109.92       0.05  24288          15         0
2232/QWE   B&D jigsaw, 8-  24-DEC-03        6     7   99.87       0.05  24288          15         1
2238/QPD   B&D cordless d  20-JAN-04       12     5   38.95       0.10  25595          12         0
23109-HB   Claw hammer     20-JAN-04       23    10    5.95       0.15  21225          25         0
23114-AA   Sledge hammer,  02-JAN-04        8    10   14.40       0.05                 12         1
54778-2T   Rat-tail file,  15-DEC-03       43    20    4.99       0.05  21344          25         0
89-WRE-Q   Hicut chain sa  07-FEB-04       11     5  256.99       0.10  24288          10         0
PVC23DRT   PVC pipe, 3.5-  20-FEB-04      188    75    5.87       0.05                 50         0
SM-18277   1.25-in. metal  01-MAR-04      172    75    6.99       0.05  21225          50         0
SW-23116   2.5-in. wd. sc  24-FEB-04      237   100    8.45       0.05  21231         100         0
WR3/TT3    Steel matting,  17-JAN-04       18     5  119.95       0.15  25595          10         0

16 rows selected.

SQL>
```

The PRC_CUS_ADD Stored Procedure

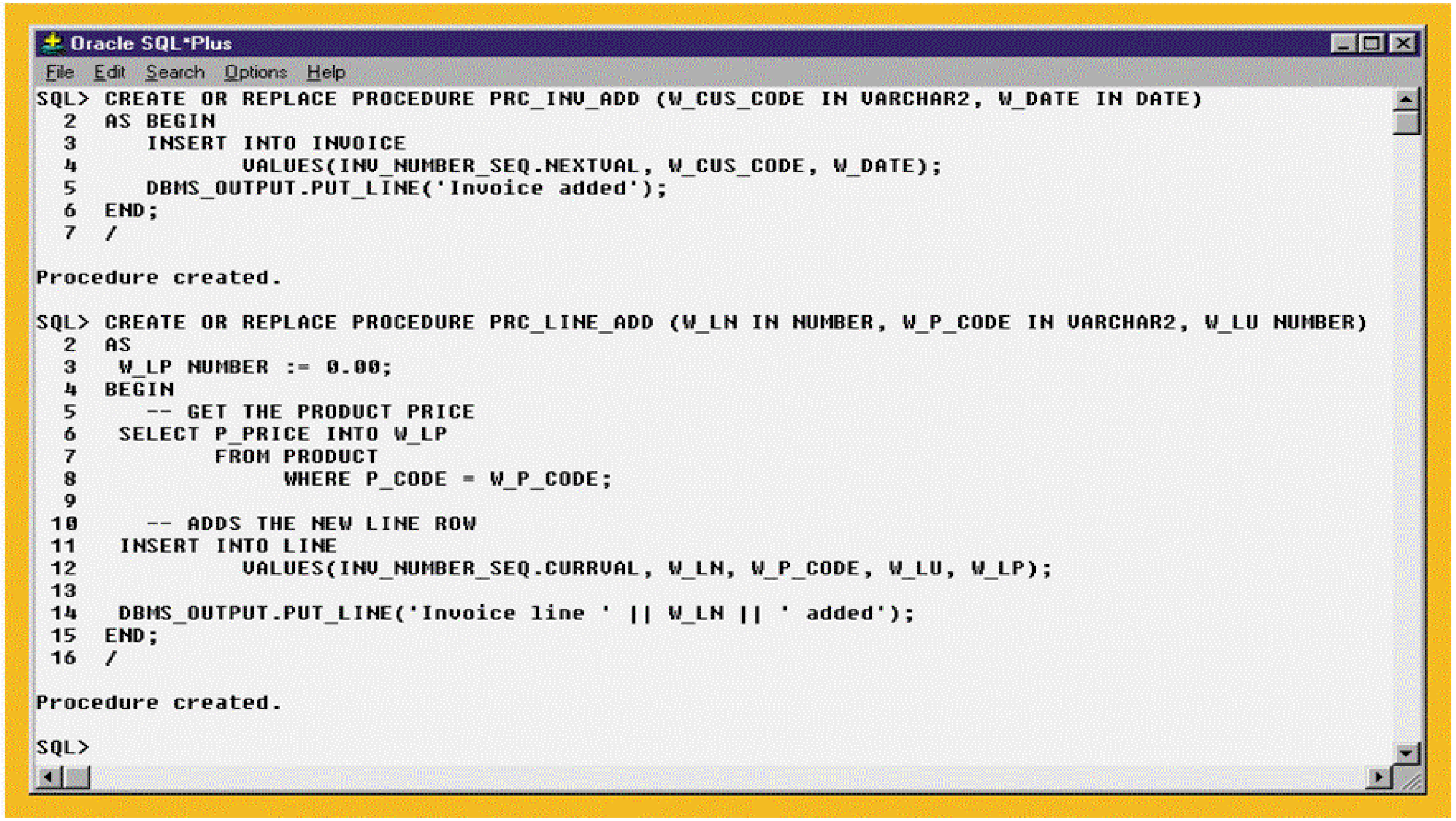
FIGURE 7.45 THE PRC_CUS_ADD STORED PROCEDURE



```
SQL> CREATE OR REPLACE PROCEDURE PRC_CUS_ADD
  2  (W_LN IN VARCHAR, W_FN IN VARCHAR, W_INIT IN VARCHAR, W_AC IN VARCHAR, W_PH IN VARCHAR)
  3  AS
  4  BEGIN
  5  -- note that the procedure uses the CUS_CODE_SEQ sequence created earlier
  6  -- attribute names are required when not giving values for all table attributes
  7     INSERT INTO CUSTOMER(CUS_CODE,CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE)
  8            VALUES (CUS_CODE_SEQ.NEXTVAL, W_LN, W_FN, W_INIT, W_AC, W_PH);
  9     DBMS_OUTPUT.PUT_LINE ('Customer ' || W_LN || ', ' || W_FN || ' added.');
 10  END;
 11  /

Procedure created.

SQL> EXEC PRC_CUS_ADD('Walker','Johnie',NULL,'615','84-DRUNK');
Customer Walker, Johnie added.

PL/SQL procedure successfully completed.

SQL> SELECT * FROM CUSTOMER WHERE CUS_LNAME = 'Walker';

  CUS_CODE CUS_LNAME       CUS_FNAME       C CUS CUS_PHON CUS_BALANCE
---------- --------------- --------------- - --- -------- -----------
     20010 Walker          Johnie            615 84-DRUNK           0

SQL> EXEC PRC_CUS_ADD('Lowery', 'Denisee', NULL, NULL, NULL);
BEGIN PRC_CUS_ADD('Lowery', 'Denisee', NULL, NULL, NULL); END;

*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("TEACHER"."CUSTOMER"."CUS_AREACODE")
ORA-06512: at "TEACHER.PRC_CUS_ADD", line 7
ORA-06512: at line 1


SQL>
```

The PRC_INV_ADD and PRC_LINE_ADD Stored Procedures

FIGURE 7.46 THE PRC_INV_ADD AND PRC_LINE_ADD STORED PROCEDURES



```
SQL> CREATE OR REPLACE PROCEDURE PRC_INV_ADD (W_CUS_CODE IN VARCHAR2, W_DATE IN DATE)
     AS BEGIN
        INSERT INTO INVOICE
               VALUES(INV_NUMBER_SEQ.NEXTVAL, W_CUS_CODE, W_DATE);
        DBMS_OUTPUT.PUT_LINE('Invoice added');
     END;
     /

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE PRC_LINE_ADD (W_LN IN NUMBER, W_P_CODE IN VARCHAR2, W_LU NUMBER)
     AS
      W_LP NUMBER := 0.00;
     BEGIN
        -- GET THE PRODUCT PRICE
      SELECT P_PRICE INTO W_LP
             FROM PRODUCT
                  WHERE P_CODE = W_P_CODE;

        -- ADDS THE NEW LINE ROW
      INSERT INTO LINE
             VALUES(INV_NUMBER_SEQ.CURRVAL, W_LN, W_P_CODE, W_LU, W_LP);

      DBMS_OUTPUT.PUT_LINE('Invoice line ' || W_LN || ' added');
     END;
     /

Procedure created.

SQL>
```

Testing the PRC_INV_ADD and PRC_LINE_ADD Procedures

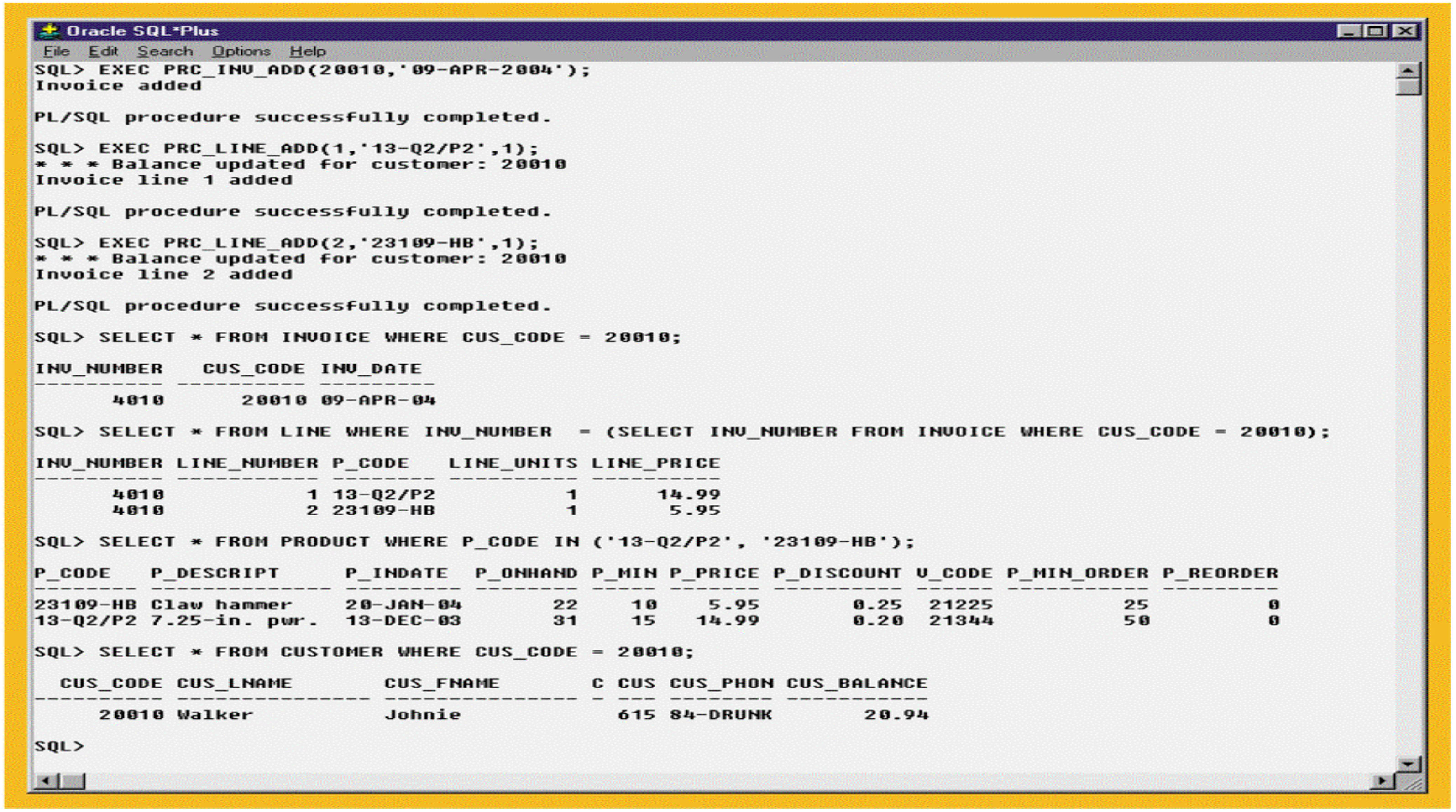
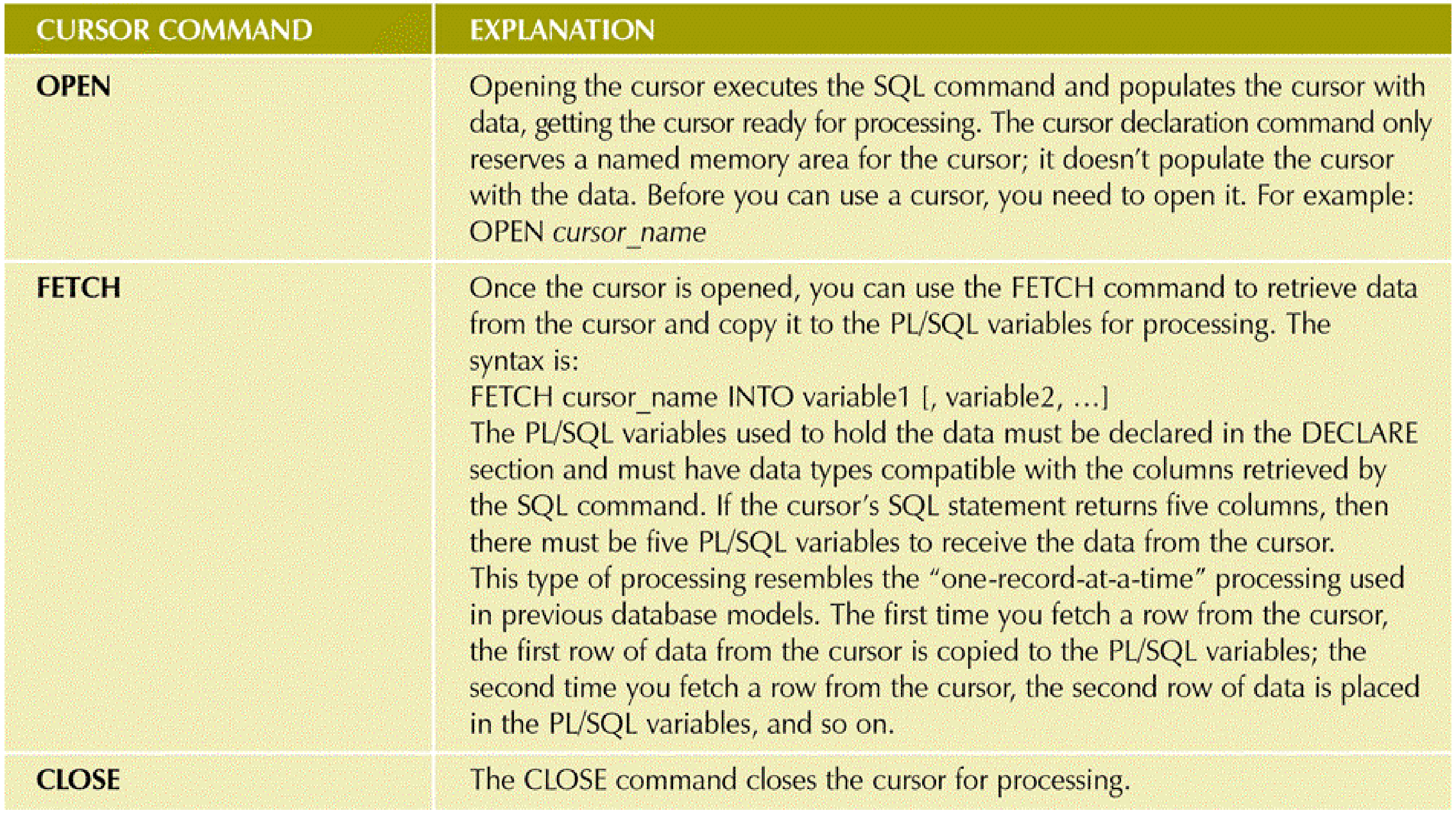


FIGURE 7.47 TESTING THE PRC_INV_ADD AND PRC_LINE_ADD PROCEDURES

```
SQL> EXEC PRC_INV_ADD(20010,'09-APR-2004');
Invoice added

PL/SQL procedure successfully completed.

SQL> EXEC PRC_LINE_ADD(1,'13-Q2/P2',1);
* * * Balance updated for customer: 20010
Invoice line 1 added

PL/SQL procedure successfully completed.

SQL> EXEC PRC_LINE_ADD(2,'23109-HB',1);
* * * Balance updated for customer: 20010
Invoice line 2 added

PL/SQL procedure successfully completed.

SQL> SELECT * FROM INVOICE WHERE CUS_CODE = 20010;

INV_NUMBER   CUS_CODE INV_DATE
---------- ---------- ---------
      4010      20010 09-APR-04

SQL> SELECT * FROM LINE WHERE INV_NUMBER  = (SELECT INV_NUMBER FROM INVOICE WHERE CUS_CODE = 20010);

INV_NUMBER LINE_NUMBER P_CODE    LINE_UNITS LINE_PRICE
---------- ----------- -------- ---------- ----------
      4010           1 13-Q2/P2          1      14.99
      4010           2 23109-HB          1       5.95

SQL> SELECT * FROM PRODUCT WHERE P_CODE IN ('13-Q2/P2', '23109-HB');

P_CODE   P_DESCRIPT     P_INDATE  P_ONHAND P_MIN P_PRICE P_DISCOUNT V_CODE P_MIN_ORDER P_REORDER
-------- -------------- --------- -------- ----- ------- ---------- ------ ----------- ---------
23109-HB Claw hammer    20-JAN-04       22    10    5.95       0.25  21225          25         0
13-Q2/P2 7.25-in. pwr.  13-DEC-03       31    15   14.99       0.20  21344          50         0

SQL> SELECT * FROM CUSTOMER WHERE CUS_CODE = 20010;

  CUS_CODE CUS_LNAME       CUS_FNAME       C CUS CUS_PHON CUS_BALANCE
---------- --------------- --------------- - --- -------- -----------
     20010 Walker          Johnie            615 84-DRUNK       20.94

SQL>
```

Cursor Processing Commands

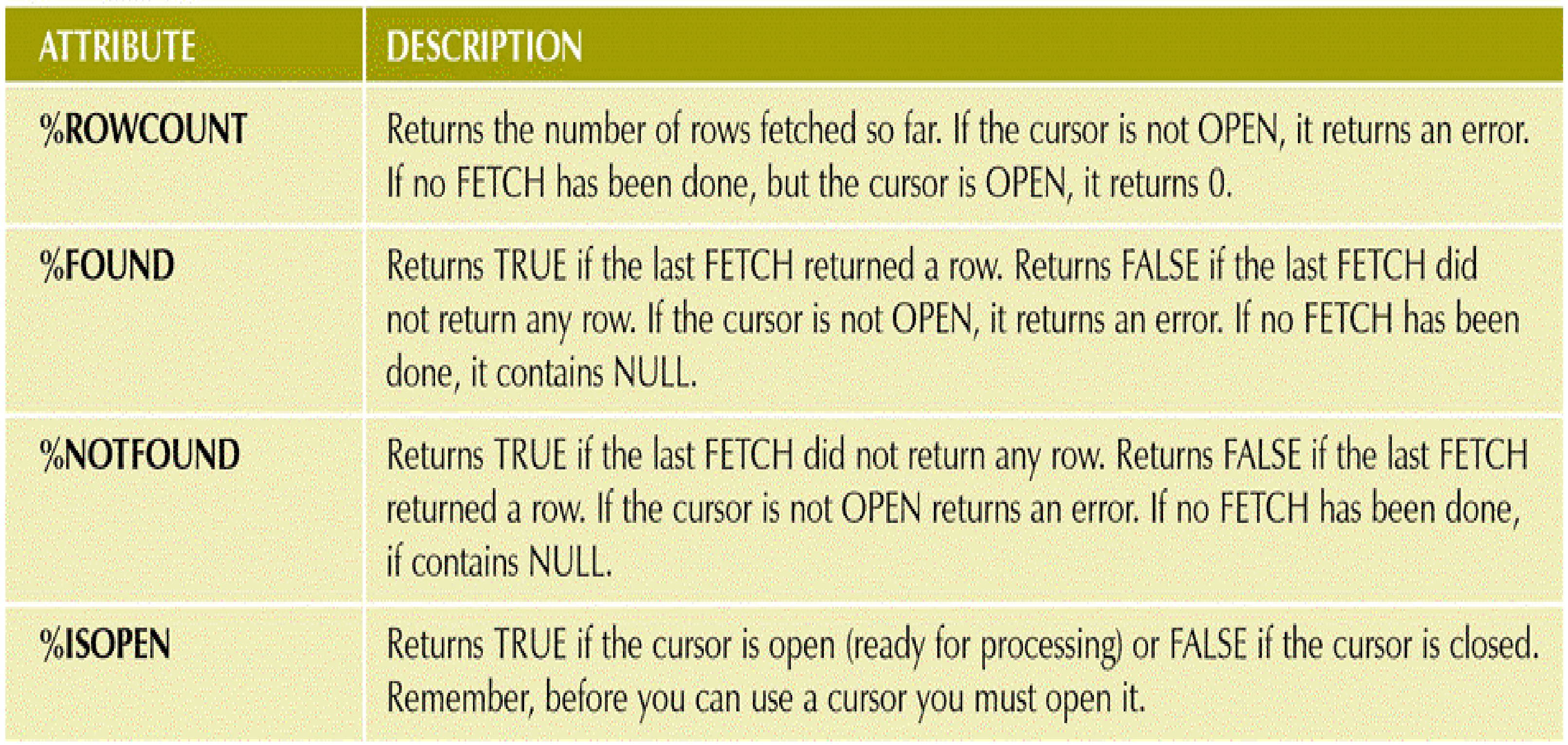
TABLE 7.9 CURSOR PROCESSING COMMANDS



CURSOR COMMAND	EXPLANATION
OPEN	Opening the cursor executes the SQL command and populates the cursor with data, getting the cursor ready for processing. The cursor declaration command only reserves a named memory area for the cursor; it doesn't populate the cursor with the data. Before you can use a cursor, you need to open it. For example: OPEN *cursor_name*
FETCH	Once the cursor is opened, you can use the FETCH command to retrieve data from the cursor and copy it to the PL/SQL variables for processing. The syntax is: FETCH cursor_name INTO variable1 [, variable2, …] The PL/SQL variables used to hold the data must be declared in the DECLARE section and must have data types compatible with the columns retrieved by the SQL command. If the cursor's SQL statement returns five columns, then there must be five PL/SQL variables to receive the data from the cursor. This type of processing resembles the "one-record-at-a-time" processing used in previous database models. The first time you fetch a row from the cursor, the first row of data from the cursor is copied to the PL/SQL variables; the second time you fetch a row from the cursor, the second row of data is placed in the PL/SQL variables, and so on.
CLOSE	The CLOSE command closes the cursor for processing.

Cursor Attributes

TABLE 7.10 CURSOR ATTRIBUTES



ATTRIBUTE	DESCRIPTION
%ROWCOUNT	Returns the number of rows fetched so far. If the cursor is not OPEN, it returns an error. If no FETCH has been done, but the cursor is OPEN, it returns 0.
%FOUND	Returns TRUE if the last FETCH returned a row. Returns FALSE if the last FETCH did not return any row. If the cursor is not OPEN, it returns an error. If no FETCH has been done, it contains NULL.
%NOTFOUND	Returns TRUE if the last FETCH did not return any row. Returns FALSE if the last FETCH returned a row. If the cursor is not OPEN returns an error. If no FETCH has been done, if contains NULL.
%ISOPEN	Returns TRUE if the cursor is open (ready for processing) or FALSE if the cursor is closed. Remember, before you can use a cursor you must open it.

SQL and Procedural Languages: Key Differences

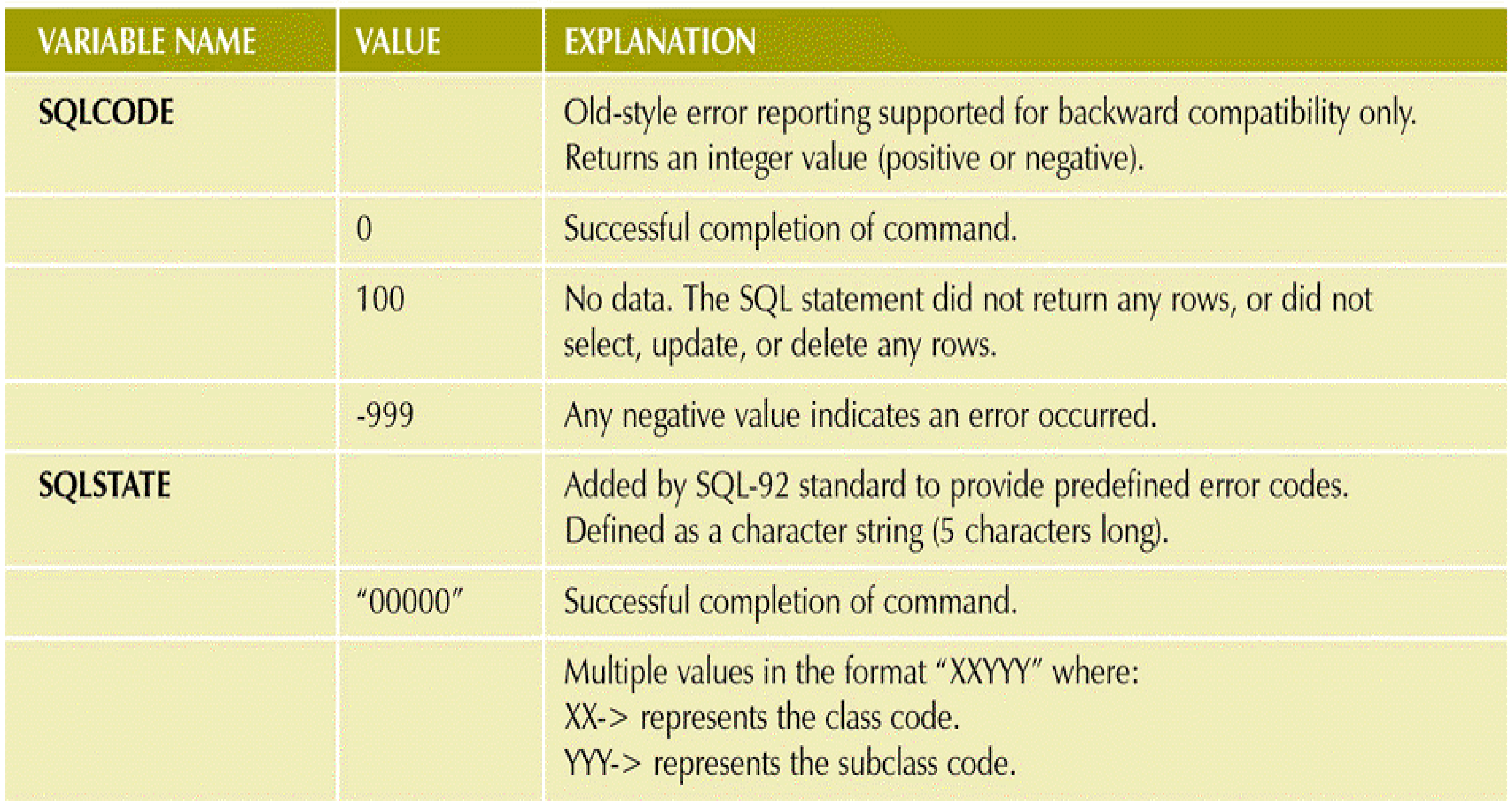
- Run-time mismatch:
  - SQL executed one instruction at a time
  - Host language typically runs at the client side in its own memory space
- Processing mismatch:
  - Host language processes one data element at a time
- Data type mismatch:
  - Data types may not match

Embedded SQL Framework

- A standard syntax to identify embedded SQL code within host language
- A standard syntax to identify host variables
- A communication area used to exchange status and error information between SQL and the host language

SQL Status and Error Reporting Variables

TABLE 7.11 SQL STATUS AND ERROR REPORTING VARIABLES



VARIABLE NAME	VALUE	EXPLANATION
SQLCODE		Old-style error reporting supported for backward compatibility only. Returns an integer value (positive or negative).
	0	Successful completion of command.
	100	No data. The SQL statement did not return any rows, or did not select, update, or delete any rows.
	-999	Any negative value indicates an error occurred.
SQLSTATE		Added by SQL-92 standard to provide predefined error codes. Defined as a character string (5 characters long).
	"00000"	Successful completion of command.
		Multiple values in the format "XXYYY" where: XX-> represents the class code. YYY-> represents the subclass code.

Static SQL

- Embedded SQL in which the programmer used predefined SQL statements and parameters
  - End users of programs are limited to actions that were specified in application programs
- SQL statements will not change while application is running

Dynamic SQL

- SQL statement is not known in advance, but instead is generated at run time
- Program can generate SQL statements at run time that are required to respond to ad hoc queries
- Attribute list and the condition are not known until the end user specifies them
- Tends to be much slower than static SQL
- Requires more computer resources

Summary

- SQL provides relational set operators to combine output of two queries to generate new relation
- Operations that join tables can be classified as inner joins and outer joins
- Subqueries and correlated queries are used when it is necessary to process data based on *other* processed data
- SQL functions are used to extract or transform data

Summary (continued)

- Oracle sequences may be used to generate values to be assigned to a record
- PL/SQL may be used to create triggers, stored procedures, and PL/SQL functions
- If SQL statements are designed to return more than one value inside the PL/SQL code, a cursor is needed
- Embedded SQL refers to the use of SQL statements within an application programming language