



Өгөгдлийн сангийн үндэс (CSII202 - 3 кр) Database Systems

Lecture 13: Advanced SQL



МУИС, ХШУИС, МКУТ-ийн багш

Маг. Довдонгийн Энхзол

Заяагүй хүн гэж язгуур угсаагүй хүнийг
хэлдэггүй юм. Заяагүй хүн гэж **залхуу,**
мунхаг хүнийг хэлдэг юм.

Ч.Лодойдамбын “Тунгалаг тамир” романаас



Outline:

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

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

Example query:

```
SELECT CUS_LNAME, CUS_FNAME,  
       CUS_INITIAL, CUS_AREACODE,  
       CUS_PHONE  
FROM CUSTOMER  
UNION  
SELECT CUS_LNAME, CUS_FNAME,  
       CUS_INITIAL, CUS_AREACODE,  
       CUS_PHONE  
FROM CUSTOMER_2;
```

UNION (continued)

FIGURE 8.1 UNION query results

The screenshot displays three database windows. The top-left window shows the 'CUSTOMER' table with 10 records. The bottom-left window shows the 'CUSTOMER_2' table with 7 records. The right window shows the results of a UNION query combining the two tables, resulting in 15 records. The UNION query window title is 'qryUNION-of-CUSTOMER-and-CUSTOMER_2 : Union Query'.

	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	vW	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	vWilliams	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

	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

	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	vW	615	894-2285
	Smith	Olette	K	615	297-3809
	Terrell	Justine	H	615	322-9870
	Tirpin	Khaleed	G	723	123-9876
	vWilliams	George		615	290-2556

Record: 1 of 15

UNION ALL

Example query:

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


UNION ALL (continued)

FIGURE
8.2

UNION ALL query results

The screenshot displays three database windows. The top-left window shows the 'CUSTOMER' table with 10 records. The bottom-left window shows the 'CUSTOMER_2' table with 7 records. The right window shows the result of a 'UNION ALL' query, which combines all records from both source tables into a single list of 17 records, maintaining their original order.

	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

	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

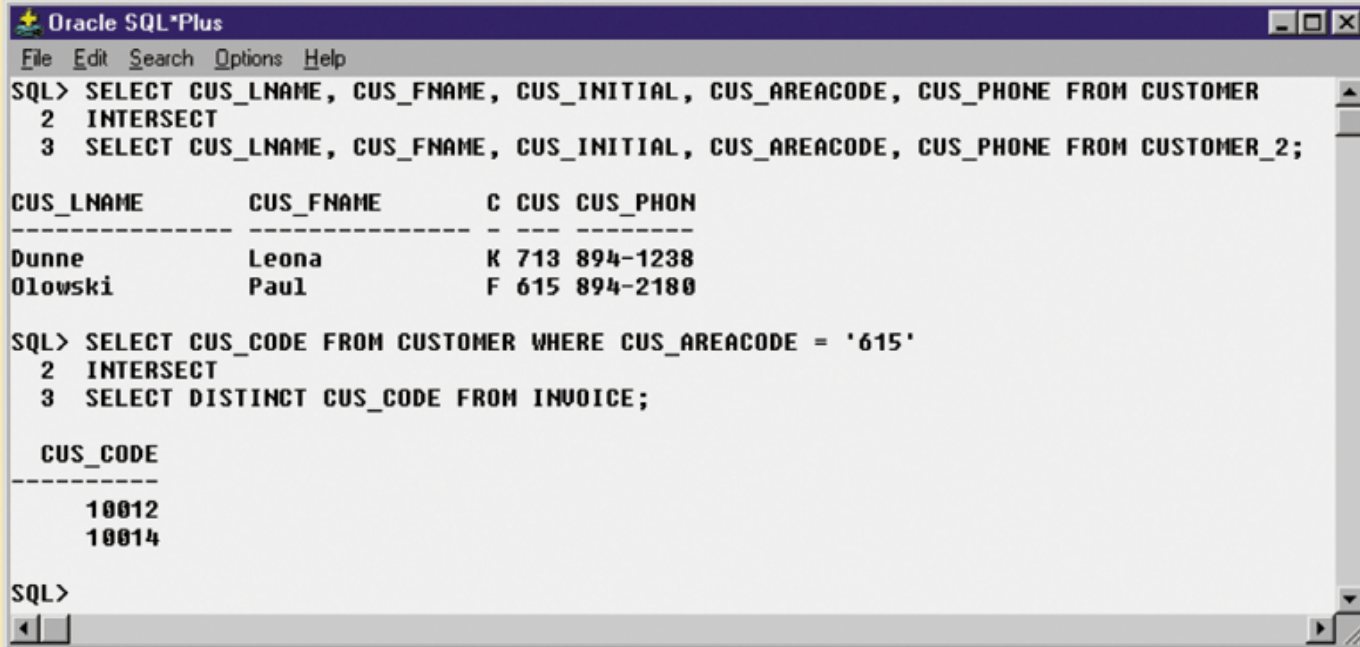
Record: 1 of 7

	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

Record: 1 of 17

INTERSECT

FIGURE 8.3 INTERSECT query results



```
Oracle SQL*Plus
File Edit Search Options Help
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

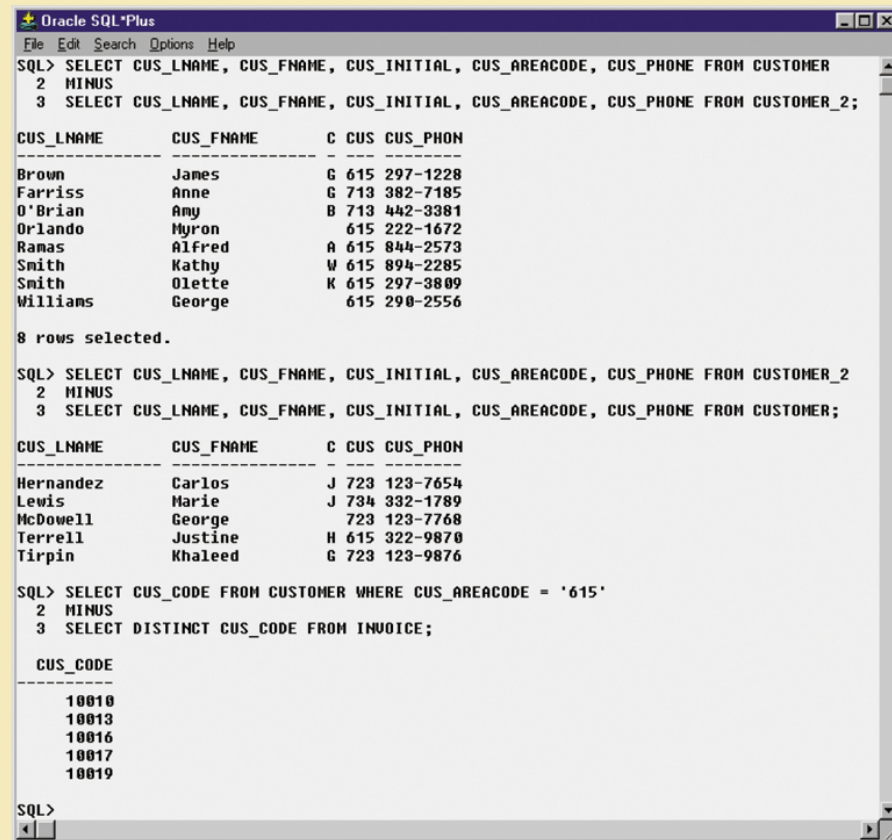
SQL> SELECT CUS_CODE FROM CUSTOMER WHERE CUS_AREACODE = '615'
2 INTERSECT
3 SELECT DISTINCT CUS_CODE FROM INVOICE;

CUS_CODE
-----
10012
10014

SQL>
```

MINUS

FIGURE 8.4 MINUS query results



The screenshot shows the Oracle SQL*Plus interface with three SQL queries and their results. The first query uses the MINUS operator to find customers in the CUSTOMER table who are not in the CUSTOMER_2 table. The second query does the opposite. The third query uses MINUS to find customer codes in the CUSTOMER table that are not in the INVOICE table.

```
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

Syntax Alternatives

FIGURE 8.5 INTERSECT alternative

The screenshot displays a database application interface with three windows. The 'CUSTOMER : Table' window shows a list of customers with columns: CUS_CODE, CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE, and CUS_BALANCE. The 'INVOICE : Table' window shows a list of invoices with columns: INV_NUMBER, CUS_CODE, and INV_DATE. The 'QRY-INTERSET-ALTERNATIVE : Sel...' window shows the results of an intersect query, listing CUS_CODE values 10012 and 10014.

	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

	INV_NUMBER	CUS_CODE	INV_DATE
+	1001	10014	16-Jan-06
+	1002	10011	16-Jan-06
+	1003	10012	16-Jan-06
+	1004	10011	17-Jan-06
+	1005	10018	17-Jan-06
+	1006	10014	17-Jan-06
+	1007	10015	17-Jan-06
+	1008	10011	17-Jan-06

Record: 1 of 8

CUS_CODE
10012
10014

Record: 1 of 2

Syntax Alternatives (continued)

FIGURE
8.6 MINUS alternative

The screenshot displays three database windows. The 'CUSTOMER : Table' window shows a list of customers with columns: CUS_CODE, CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE, CUS_PHONE, and CUS_BALANCE. The 'INVOICE : Table' window shows a list of invoices with columns: INV_NUMBER, CUS_CODE, and INV_DATE. The 'qry-MINUS-ALTERNATIVE : Select Query' window shows the result of a MINUS query, listing customer codes that are in the CUSTOMER table but not in the INVOICE 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

	INV_NUMBER	CUS_CODE	INV_DATE
+	1001	10014	16-Jan-06
+	1002	10011	16-Jan-06
+	1003	10012	16-Jan-06
+	1004	10011	17-Jan-06
+	1005	10018	17-Jan-06
+	1006	10014	17-Jan-06
+	1007	10015	17-Jan-06
+	1008	10011	17-Jan-06

	CUS_CODE
+	10010
+	10013
+	10016
+	10017
+	10019

SQL Join Operators

TABLE
8.1

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.

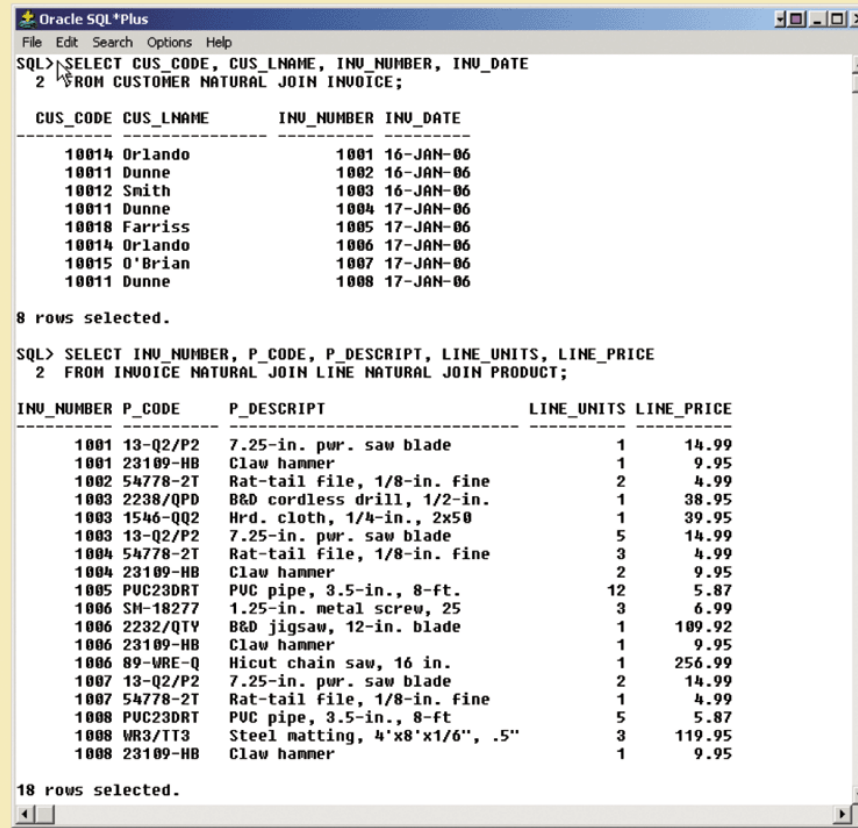
Cross Join

Syntax:

```
SELECT column-list FROM table1 CROSS JOIN  
table2
```

Natural Join

FIGURE 8.7
NATURAL JOIN results



The screenshot shows the Oracle SQL*Plus interface. The first query is a NATURAL JOIN between the CUSTOMER and INVOICE tables, resulting in 8 rows. The second query is a NATURAL JOIN between the INVOICE, LINE, and PRODUCT tables, resulting in 18 rows.

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-06
10011	Dunne	1002	16-JAN-06
10012	Smith	1003	16-JAN-06
10011	Dunne	1004	17-JAN-06
10018	Farriss	1005	17-JAN-06
10014	Orlando	1006	17-JAN-06
10015	O'Brian	1007	17-JAN-06
10011	Dunne	1008	17-JAN-06

8 rows selected.

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

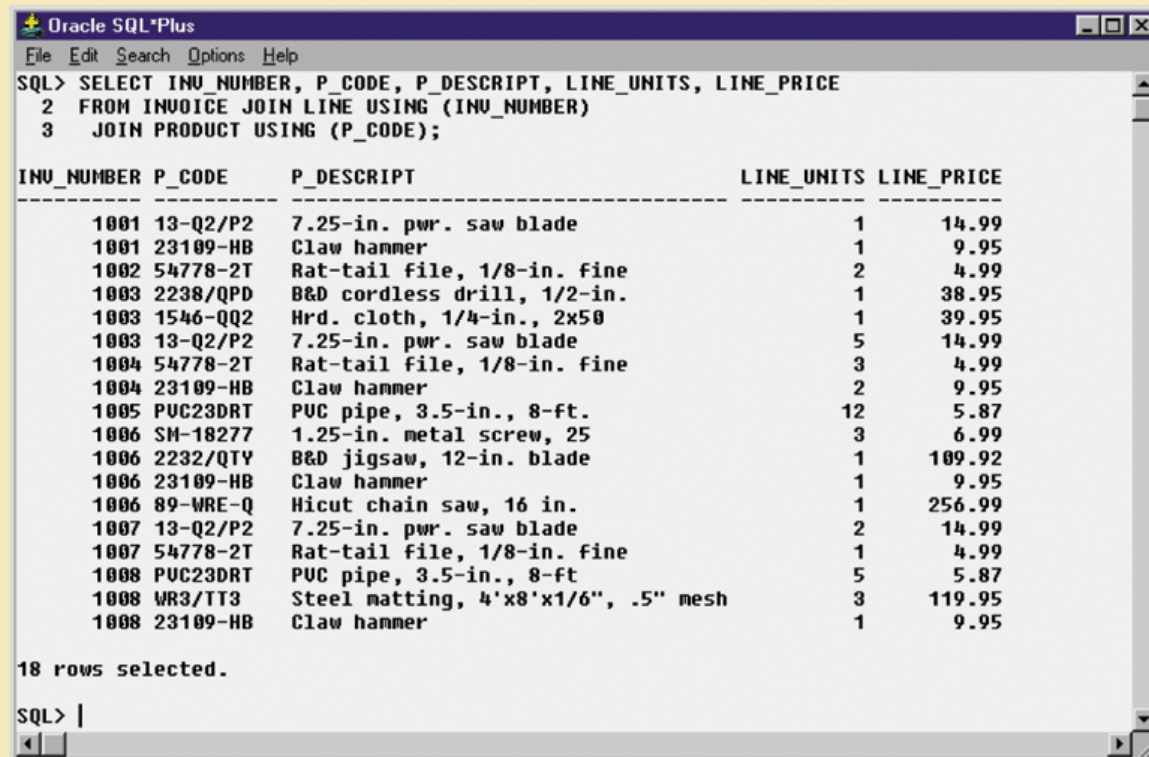
INV_NUMBER	P_CODE	P_DESCRIPT	LINE_UNITS	LINE_PRICE
1001	13-Q2/P2	7.25-in. pwr. saw blade	1	14.99
1001	23109-HB	Claw hammer	1	9.95
1002	54778-2T	Rat-tail file, 1/8-in. fine	2	4.99
1003	2238/QPD	B&D cordless drill, 1/2-in.	1	38.95
1003	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	1	39.95
1003	13-Q2/P2	7.25-in. pwr. 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	PUC230RT	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	PUC230RT	PVC pipe, 3.5-in., 8-ft	5	5.87
1008	WR3/TT3	Steel matting, 4'x8'x1/6", .5"	3	119.95
1008	23109-HB	Claw hammer	1	9.95

18 rows selected.

JOIN USING Clause

FIGURE
8.8

JOIN USING results



The screenshot shows the Oracle SQL*Plus interface. The command window contains the following SQL query:

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

The results are displayed in a table with the following columns: INU_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, and LINE_PRICE. There are 18 rows of data.

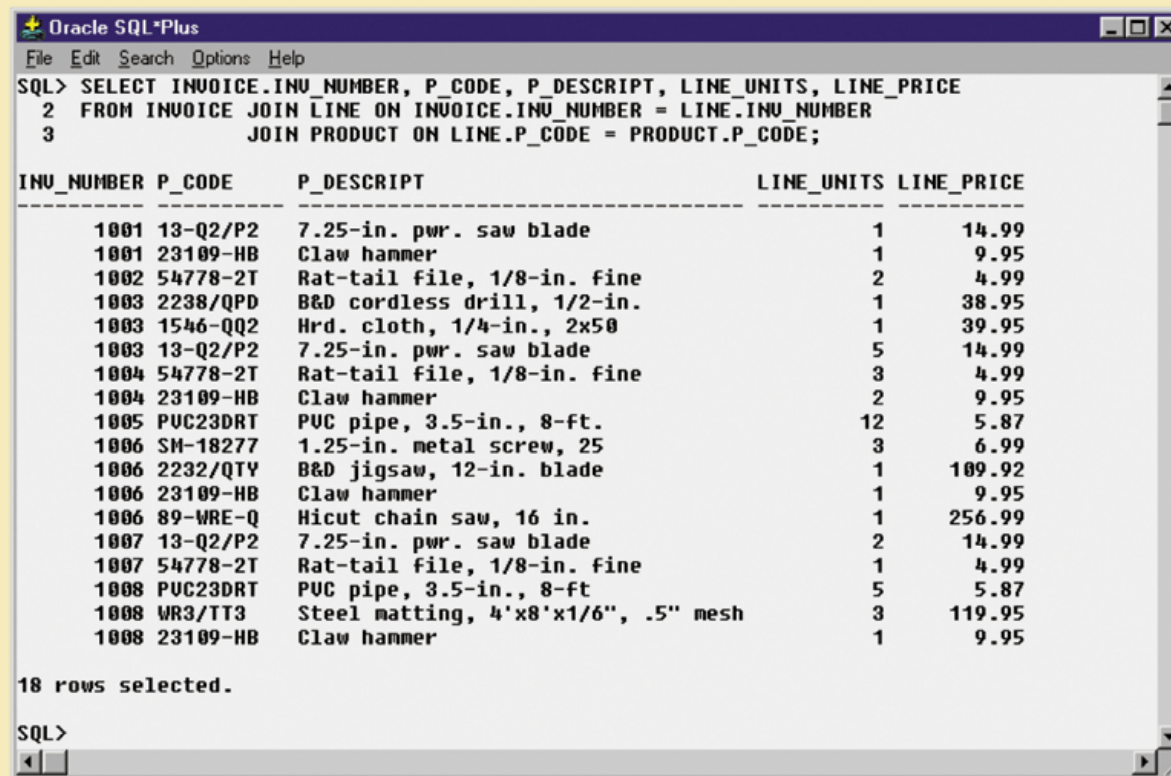
INU_NUMBER	P_CODE	P_DESCRIPT	LINE_UNITS	LINE_PRICE
1001	13-Q2/P2	7.25-in. pwr. saw blade	1	14.99
1001	23109-HB	Claw hammer	1	9.95
1002	54778-2T	Rat-tail file, 1/8-in. fine	2	4.99
1003	2238/QPD	B&D cordless drill, 1/2-in.	1	38.95
1003	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	1	39.95
1003	13-Q2/P2	7.25-in. pwr. 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 Clause

FIGURE 8.9 JOIN ON results



The screenshot shows an Oracle SQL*Plus window with a menu bar (File, Edit, Search, Options, Help) and a command area. The command entered is:

```
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;
```

The results are displayed in a table with the following columns: INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, and LINE_PRICE. There are 18 rows of data.

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/QT	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 matching rows, but also rows with unmatched attribute values for one table or both tables to be joined

Three types

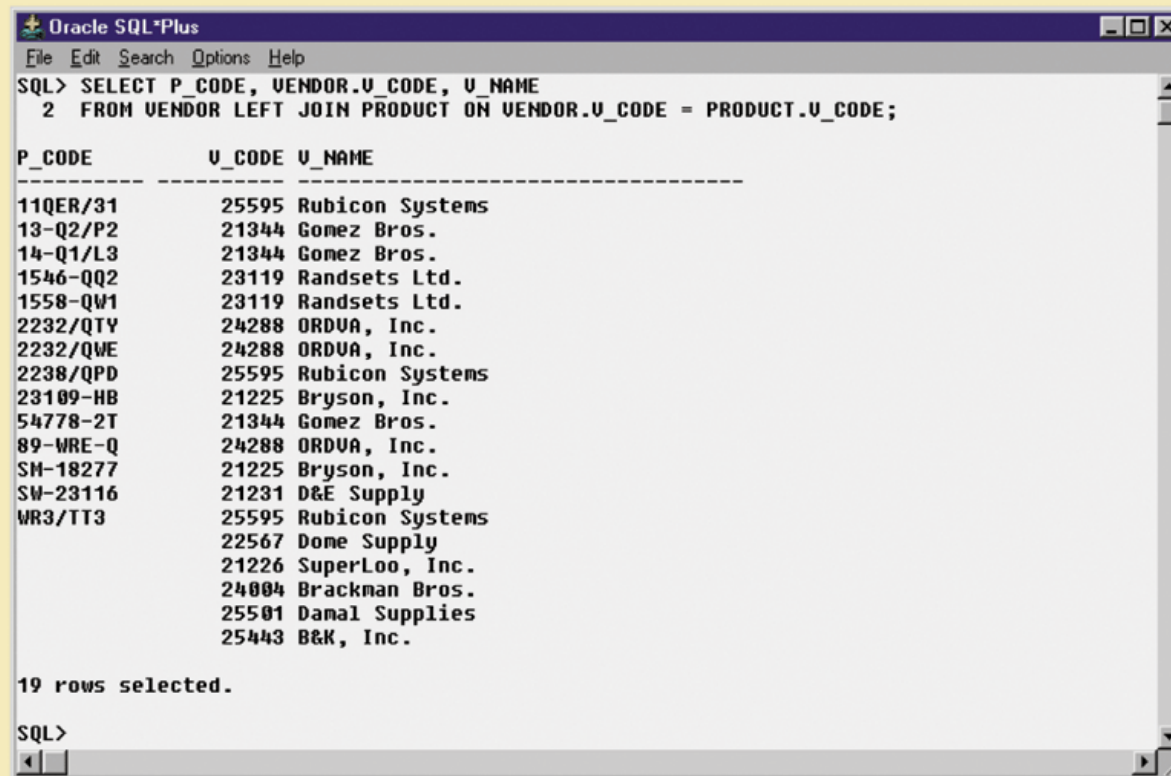
Left

Right

Full

Outer Joins (continued)

FIGURE 8.10 LEFT JOIN results



The screenshot shows the Oracle SQL*Plus interface. The command window contains the following SQL query:

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

The results are displayed in a table with three columns: P_CODE, V_CODE, and V_NAME. There are 19 rows selected.

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/QTU	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.

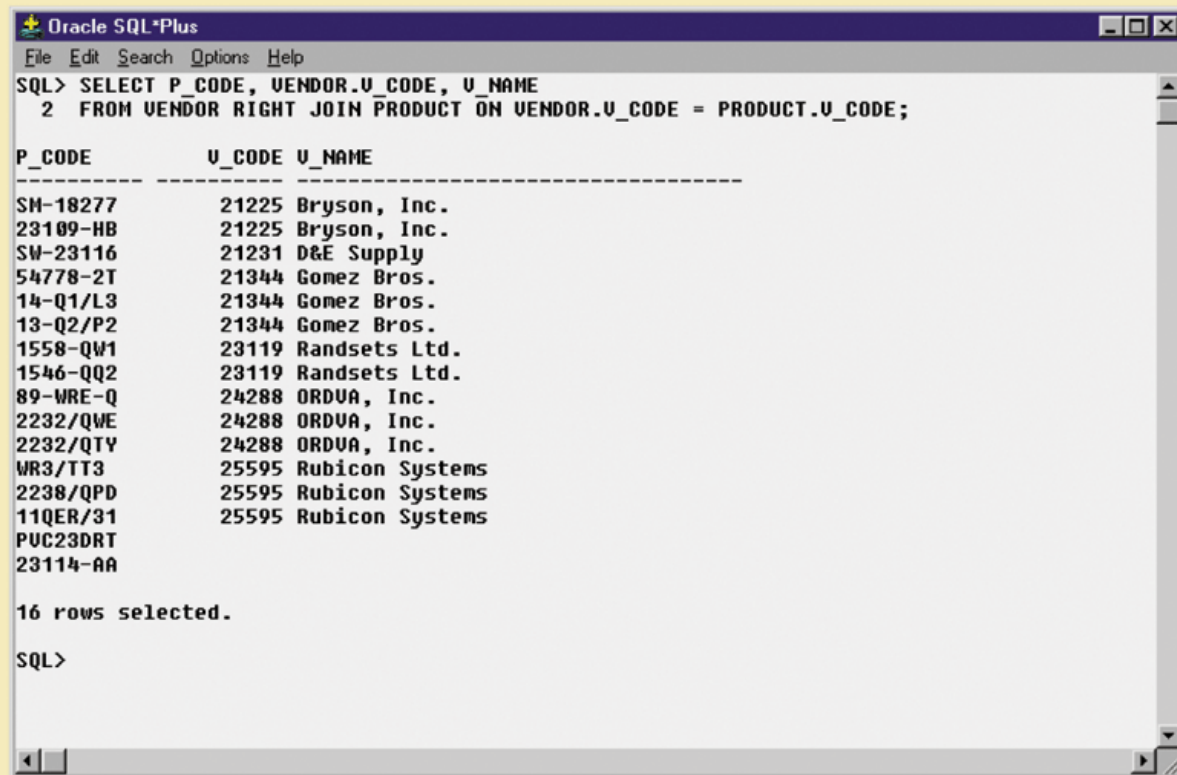
19 rows selected.

SQL>

Outer Joins (continued)

FIGURE
8.11

RIGHT JOIN results



The screenshot shows the Oracle SQL*Plus interface. The command window contains the following SQL query:

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

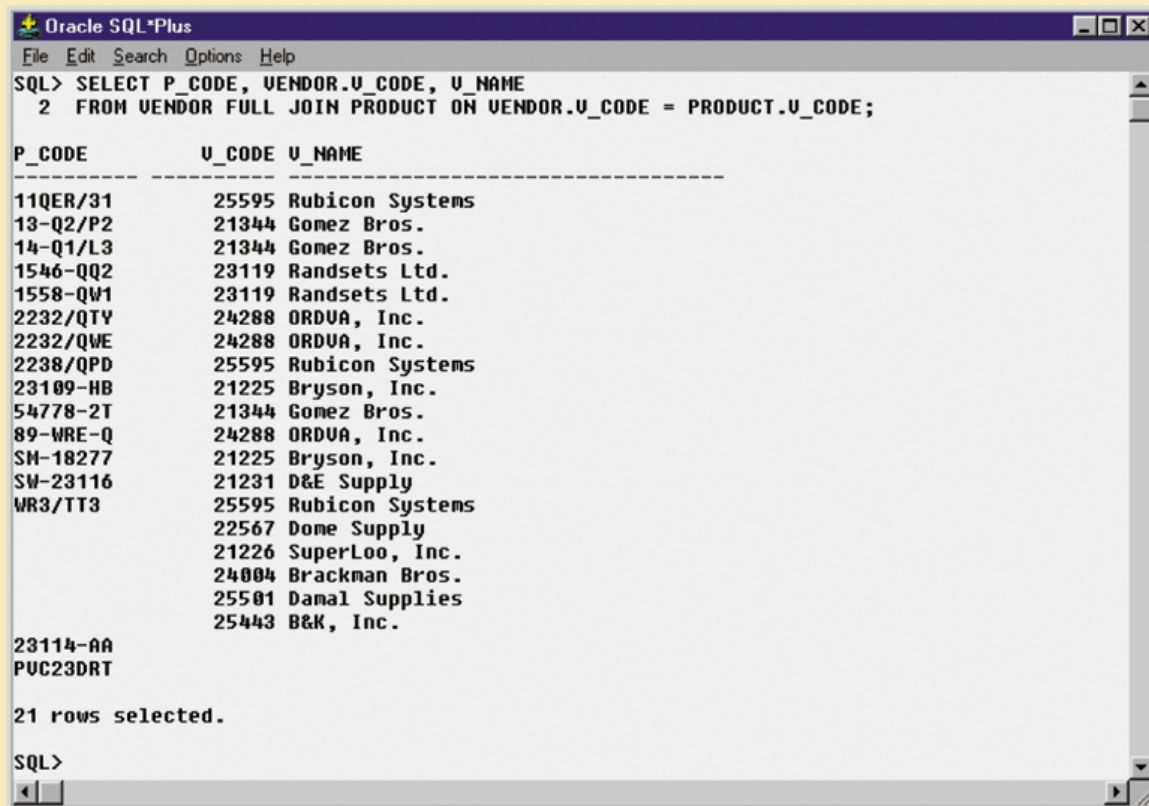
The results are displayed in a table with three columns: P_CODE, V_CODE, and V_NAME. The table shows 16 rows of data, representing products that have a corresponding vendor. The data is as follows:

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/QTU	24288	ORDVA, Inc.
WR3/TT3	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
11QER/31	25595	Rubicon Systems
PUC23DRT		
23114-AA		

Below the table, the message "16 rows selected." is displayed. The prompt "SQL>" is visible at the bottom of the command window.

Outer Joins (continued)

FIGURE 8.12 FULL JOIN results



The screenshot shows the Oracle SQL*Plus interface with a query window. The query is a FULL JOIN between the VENDOR and PRODUCT tables on the V_CODE column. The results are displayed in a table with three columns: P_CODE, V_CODE, and V_NAME. The table lists 21 rows of data, showing various product codes and their corresponding vendor information. The window title is 'Oracle SQL*Plus' and it includes standard menu options like File, Edit, Search, Options, and Help.

```
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/QTU	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.
SH-18277	21225	Bryson, Inc.
SW-23116	21231	D&E Supply
WR3/TT3	25595	Rubicon Systems
	22567	Dome Supply
	21226	SuperLoe, Inc.
	24004	Brackman Bros.
	25501	Damal Supplies
	25443	B&K, Inc.
23114-AA		
PUC23DRT		

21 rows selected.

```
SQL>
```


Subqueries and Correlated Queries

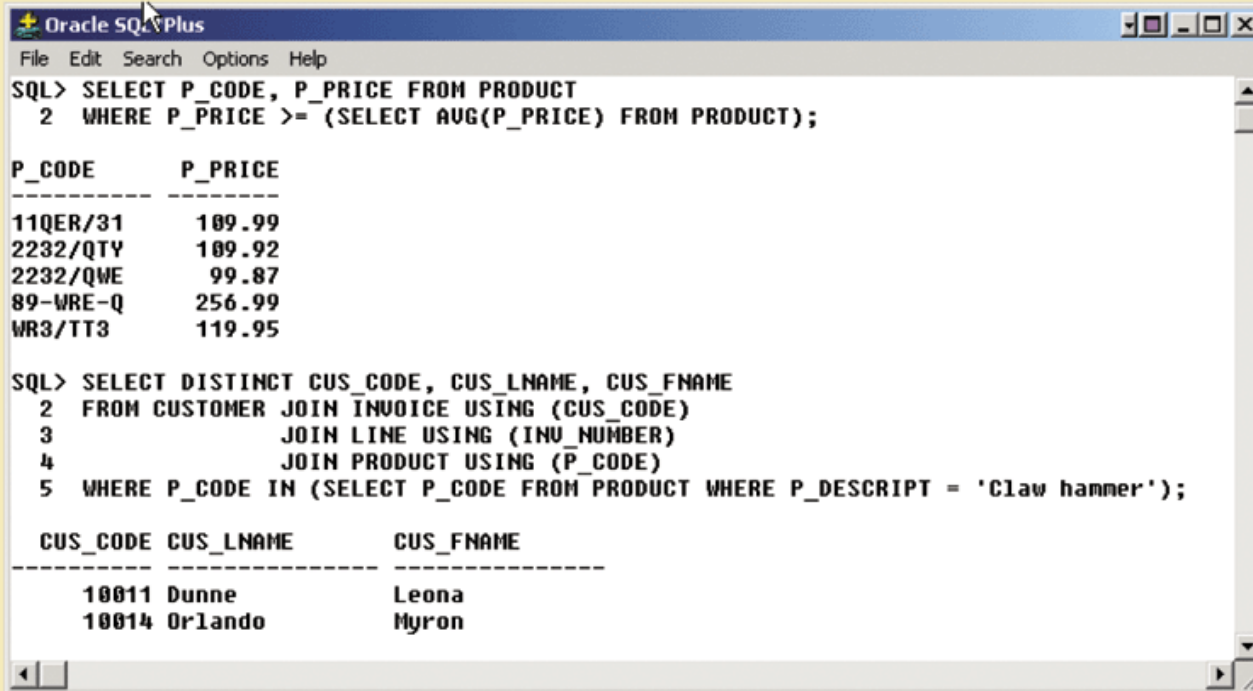
TABLE 8.2 SELECT Subquery Examples

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.

WHERE Subqueries

FIGURE
8.13

WHERE subquery examples



The screenshot shows the Oracle SQL*Plus interface with two SQL queries and their results. The first query filters products by price, and the second query joins customer, invoice, and product tables to find customers who bought claw hammers.

```
Oracle SQL*Plus
File Edit Search Options Help

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

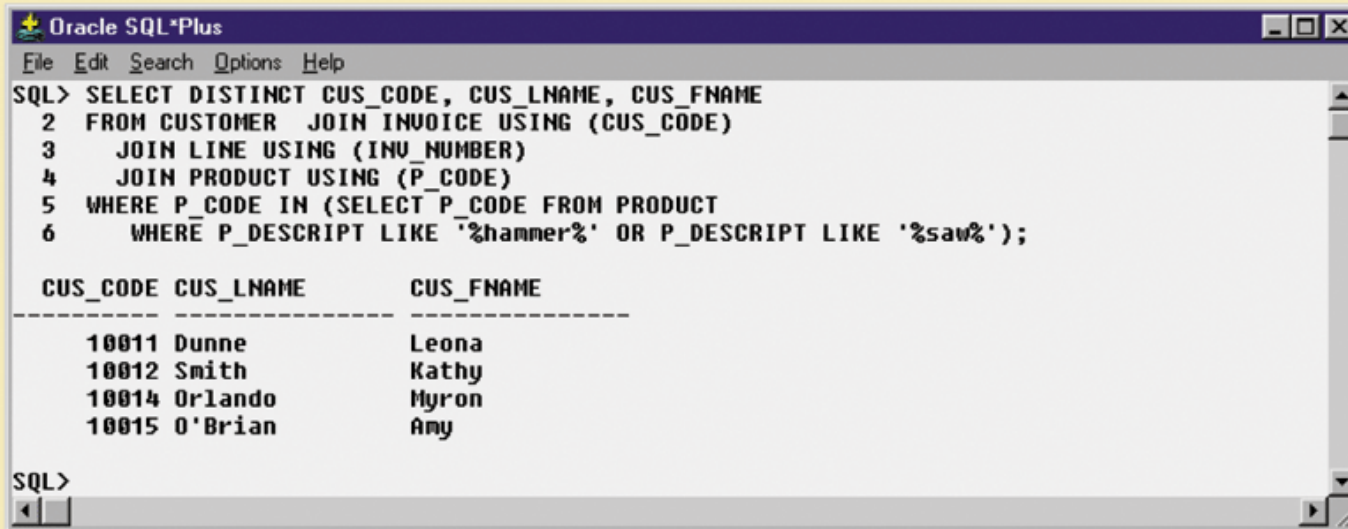
```
SQL> SELECT DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
2  FROM CUSTOMER JOIN INVOICE USING (CUS_CODE)
3                JOIN LINE USING (INV_NUMBER)
4                JOIN PRODUCT USING (P_CODE)
5  WHERE P_CODE IN (SELECT P_CODE FROM PRODUCT WHERE P_DESCRIPT = 'Claw hammer');
```

CUS_CODE	CUS_LNAME	CUS_FNAME
10011	Dunne	Leona
10014	Orlando	Myron

IN Subqueries

FIGURE
8.14

IN subquery examples



```
Oracle SQL*Plus
File Edit Search Options Help
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 IN (SELECT P_CODE FROM PRODUCT
6  WHERE P_DESCRIPT LIKE '%hammer%' OR P_DESCRIPT LIKE '%saw%');

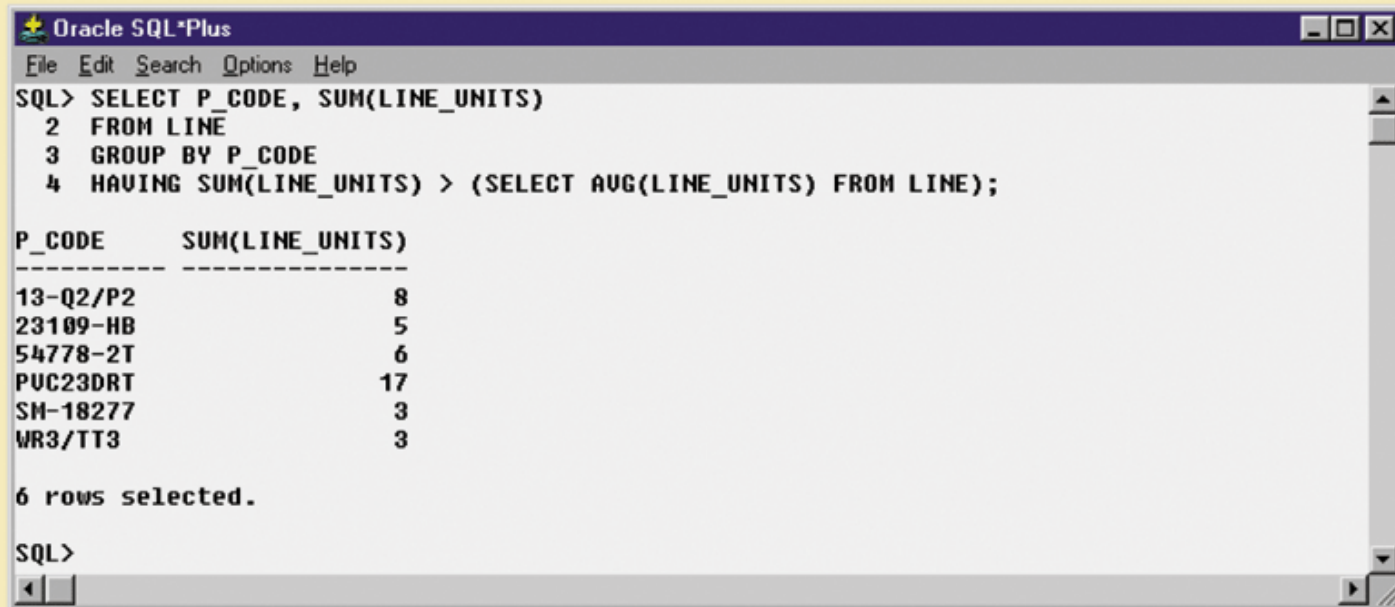
CUS_CODE CUS_LNAME      CUS_FNAME
-----
10011 Dunne         Leona
10012 Smith         Kathy
10014 Orlando       Myron
10015 O'Brian       Amy

SQL>
```

HAVING Subqueries

FIGURE
8.15

HAVING subquery examples



The screenshot shows the Oracle SQL*Plus interface. The title bar reads "Oracle SQL*Plus". The menu bar includes "File", "Edit", "Search", "Options", and "Help". The command window contains the following SQL query:

```
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);
```

The query results are displayed in a table with two columns: "P_CODE" and "SUM(LINE_UNITS)". The results are as follows:

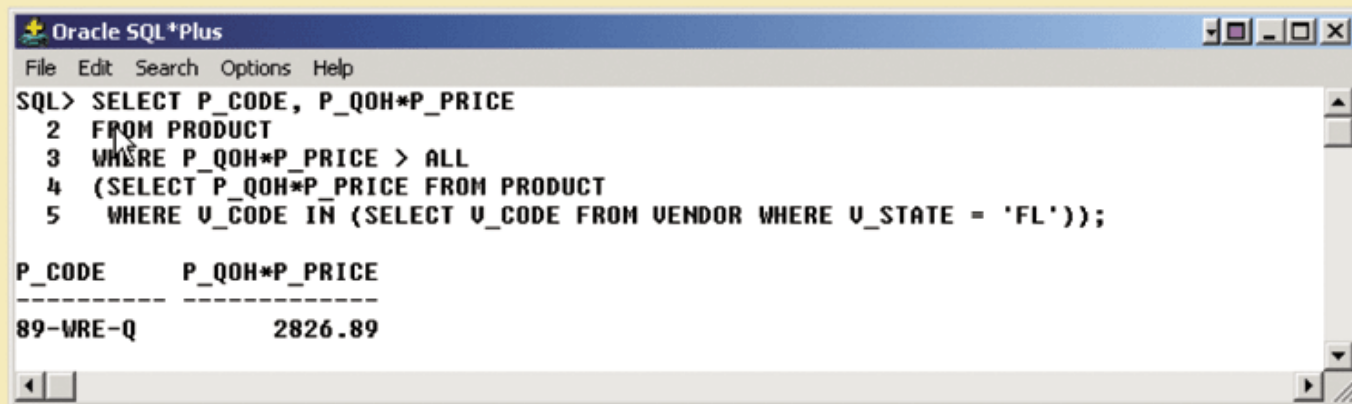
P_CODE	SUM(LINE_UNITS)
13-Q2/P2	8
23109-HB	5
54778-2T	6
PUC23DRT	17
SH-18277	3
WR3/TT3	3

Below the table, the text "6 rows selected." is displayed. The prompt "SQL>" is visible at the bottom of the command window.

Multirow Subquery Operators: ANY and ALL

FIGURE
8.16

Multirow subquery operator example



The screenshot shows the Oracle SQL*Plus interface. The command window contains the following SQL query:

```
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'));
```

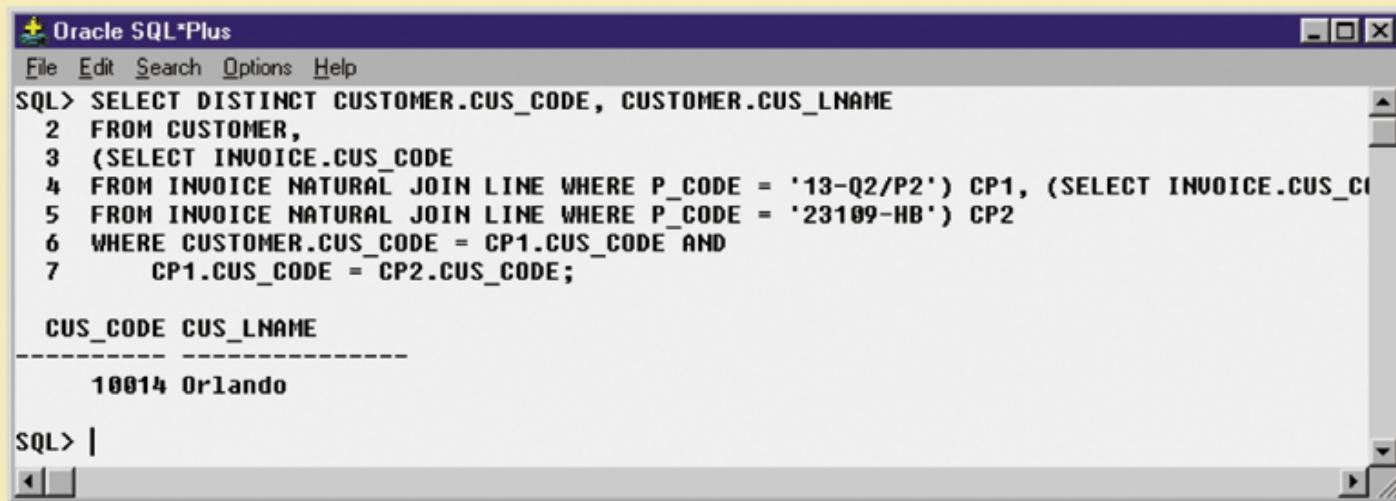
The query result is displayed in a table format:

P_CODE	P_QOH*P_PRICE
89-WRE-Q	2826.89

FROM Subqueries

FIGURE
8.17

FROM subquery example



```
Oracle SQL*Plus
File Edit Search Options Help
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_CODE
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;

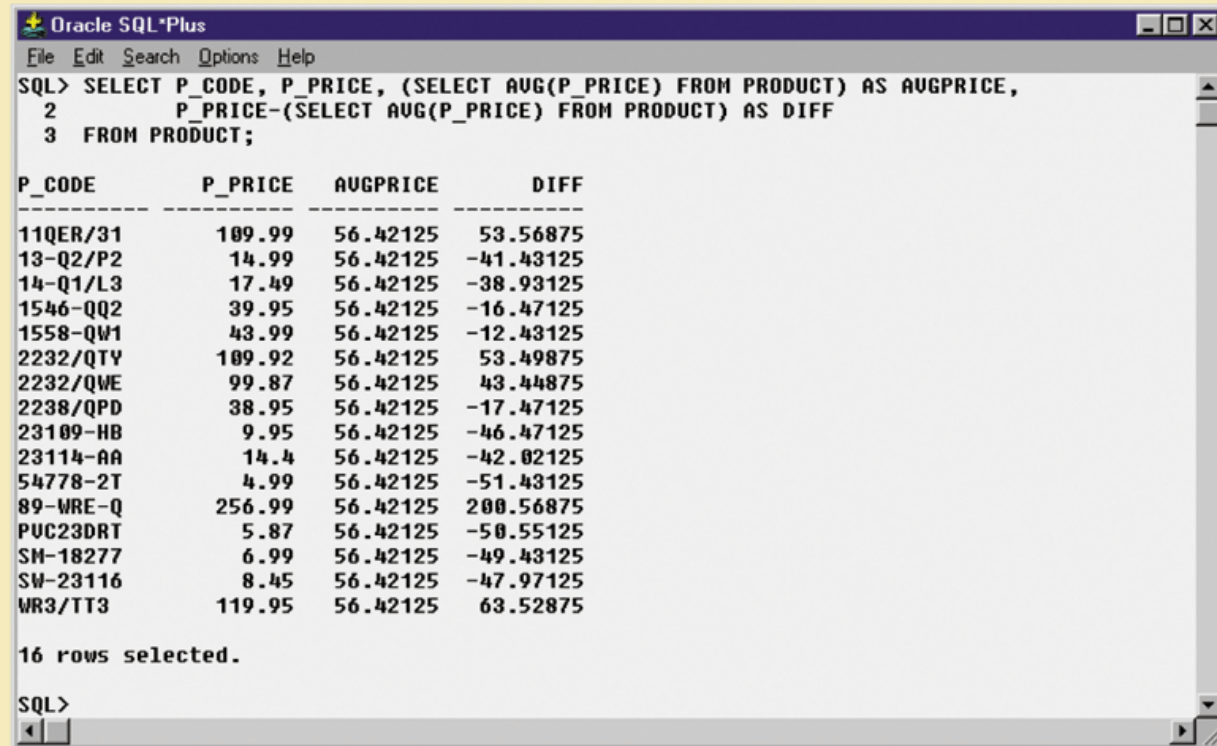
CUS_CODE CUS_LNAME
-----
10014 Orlando

SQL> |
```

Attribute List Subqueries

FIGURE
8.18

Inline subquery example



The screenshot shows the Oracle SQL*Plus interface. The command window contains the following SQL query:

```
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;
```

The result set displays 16 rows of data with the following columns: P_CODE, P_PRICE, AVGPRICE, and DIFF. The AVGPRICE column shows a constant value of 56.42125 for all rows, and the DIFF column shows the difference between the product's price and this average.

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
SH-18277	6.99	56.42125	-49.43125
SW-23116	8.45	56.42125	-47.97125
WR3/TT3	119.95	56.42125	63.52875

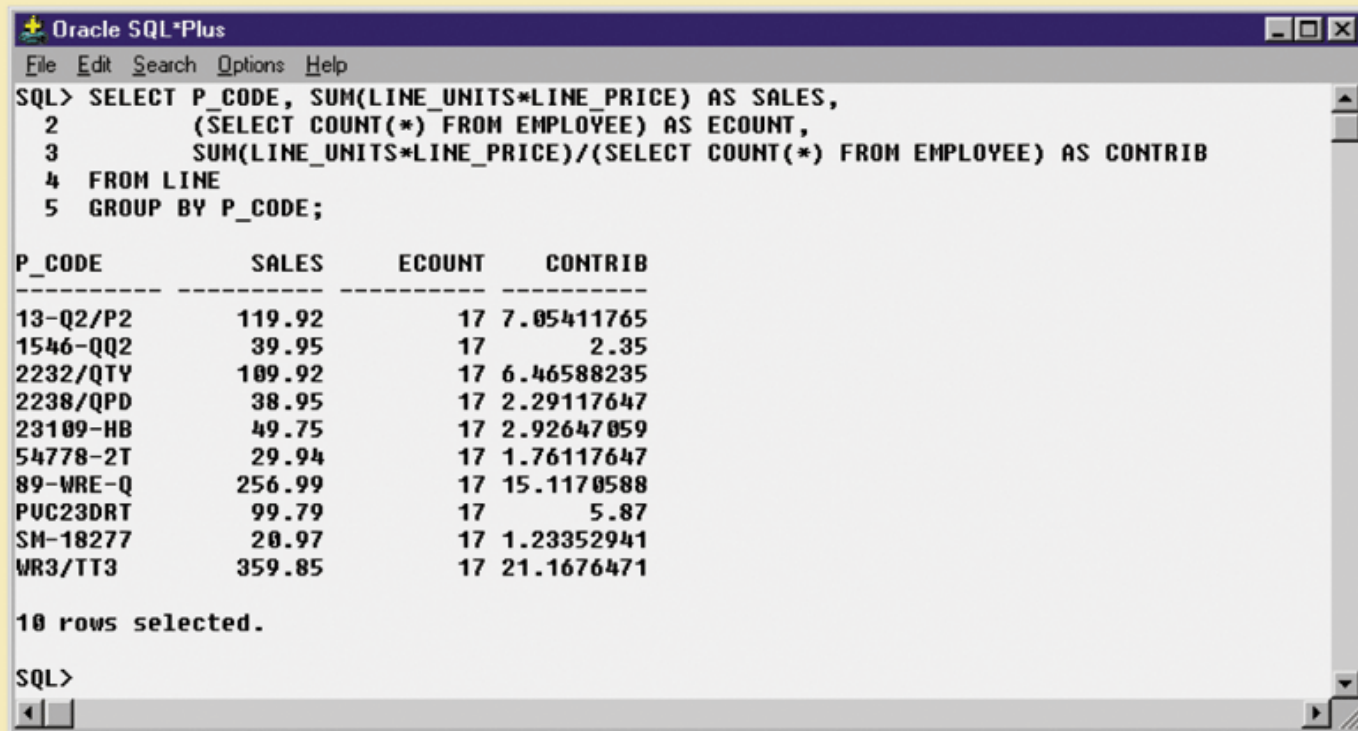
16 rows selected.

SQL>

Attribute List Subqueries (continued)

FIGURE
8.19

Another example of an inline subquery



```
Oracle SQL*Plus
File Edit Search Options Help
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
PUC23DRT	99.79	17	5.87
SM-18277	20.97	17	1.23352941
WR3/TT3	359.85	17	21.1676471

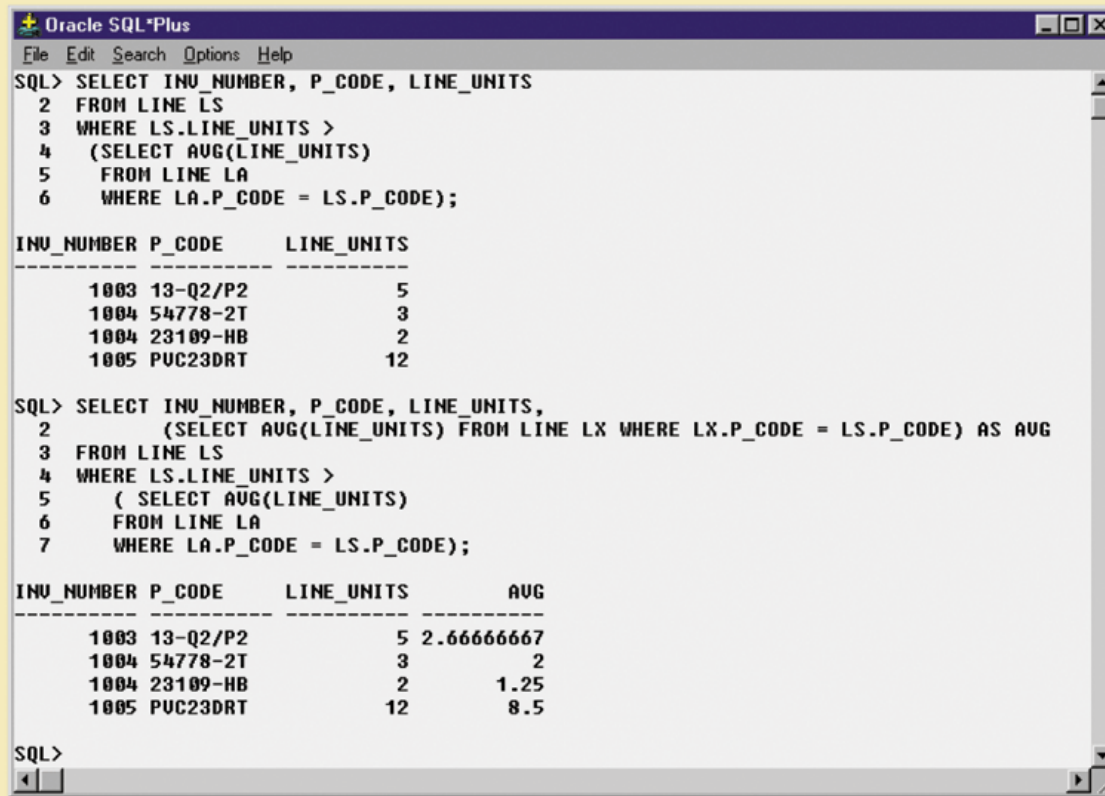
10 rows selected.

```
SQL>
```

Correlated Subqueries

FIGURE
8.20

Correlated subquery examples



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT INU_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);
```

INU_NUMBER	P_CODE	LINE_UNITS
1003	13-Q2/P2	5
1004	54778-2T	3
1004	23109-HB	2
1005	PUC23DRT	12

```
SQL> SELECT INU_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);
```

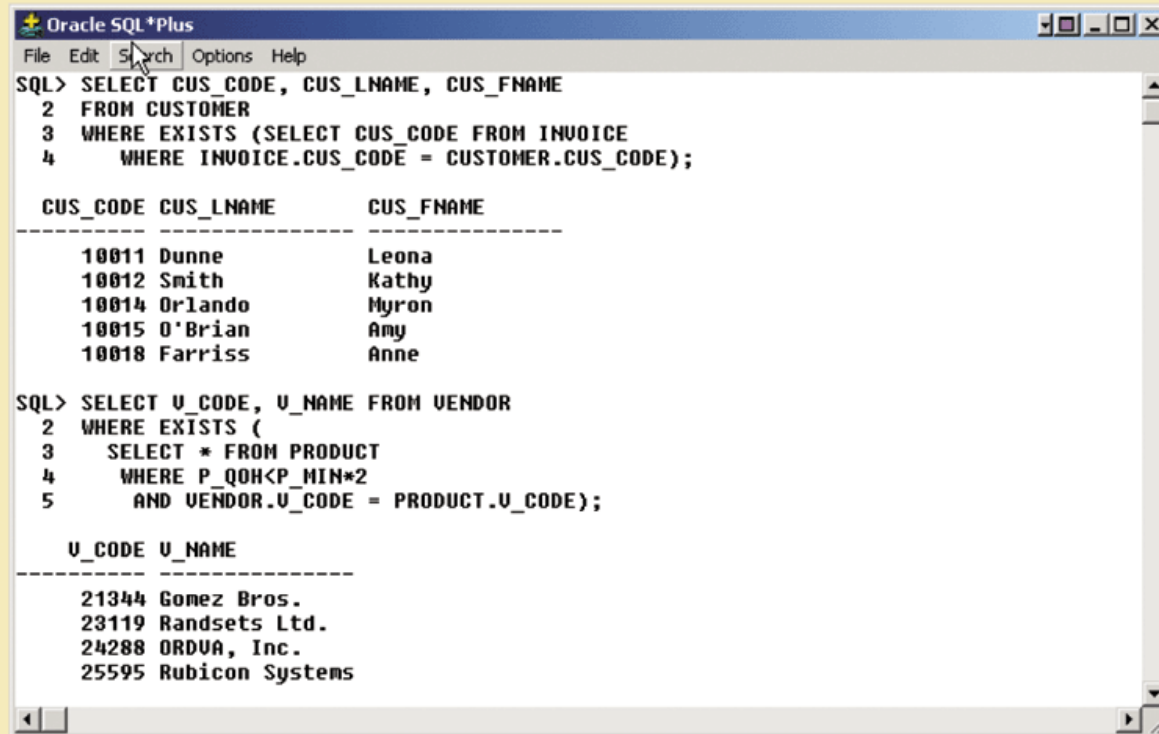
INU_NUMBER	P_CODE	LINE_UNITS	AVG
1003	13-Q2/P2	5	2.66666667
1004	54778-2T	3	2
1004	23109-HB	2	1.25
1005	PUC23DRT	12	8.5

```
SQL>
```


Correlated Subqueries (continued)

FIGURE
8.21

EXISTS correlated subquery examples



The screenshot shows the Oracle SQL*Plus interface with two SQL queries and their results. The first query selects customers with invoices, and the second query selects vendors with products.

```
Oracle SQL*Plus
File Edit Search Options Help
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 U_CODE, U_NAME FROM VENDOR
2 WHERE EXISTS (
3 SELECT * FROM PRODUCT
4 WHERE P_QOH < P_MIN * 2
5 AND VENDOR.U_CODE = PRODUCT.U_CODE);
```

U_CODE	U_NAME
21344	Gomez Bros.
23119	Randsets Ltd.
24288	ORDVA, Inc.
25595	Rubicon Systems

Date and Time Functions

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: <ul style="list-style-type: none">• 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	<p>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';</p> <p>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';</p> <p>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';</p>

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): <pre>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;</pre> <p>Note the following:</p> <ul style="list-style-type: none"> '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. <p>How many days between Thanksgiving and Christmas 2006?</p> <pre>SELECT TO_DATE('2006/12/25','YYYY/MM/DD') - TO_DATE('NOVEMBER 23, 2006','MONTH DD, YYYY') FROM DUAL;</pre> <p>Note the following:</p> <ul style="list-style-type: none"> 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: <pre>SELECT TO_DATE('25-Dec-2006','DD-MON-YYYY') - SYSDATE FROM DUAL;</pre> <p>Notice two things:</p> <ul style="list-style-type: none"> 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): <pre>SELECT P_CODE, P_INDATE, ADD_MONTHS(P_INDATE,24) FROM PRODUCT ORDER BY ADD_MONTHS(P_INDATE,24);</pre>
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: <pre>SELECT EMP_LNAME, EMP_FNAME, EMP_HIRE_DATE FROM EMPLOYEE WHERE EMP_HIRE_DATE >= LAST_DAY(EMP_HIRE_DATE)-7;</pre>