

Database Systems: Design, Implementation, and Management

UNIT 4
Advanced SQL

SQL Join Operators

- Join operation merges rows from two tables and returns the rows with one of the following:
 - *Inner Join*
 - Have common values in common columns
 - Natural join
 - Meet a given join condition
 - Equality or inequality
 - *Outer join*
 - Have common values in common columns or have no matching value
 - *Cross join*
 - Returns same result as the Cartesian product of two sets

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

- Performs relational product of two tables
 - Also called Cartesian product
- Syntax:
 - SELECT column-list FROM table1 CROSS JOIN table2
- Perform a cross join that yields specified attributes

```
SELECT INVOICE.INV_NUMBER, CUS_CODE, INV_DATE, P_CODE  
FROM INVOICE CROSS JOIN LINE;
```

OR

```
FROM INVOICE, LINE;
```

Natural Join

- Not available in MS SQL – use JOIN ON
- Returns all rows with matching values in the matching columns
 - Determines the common attribute(s) by looking for attributes with identical names and compatible data types
 - Select only the rows with common values in the common attribute(s)
 - If there are no common attributes, return the relational product of the two tables
 - Eliminates duplicate columns
- Used when tables share one or more common attributes with common names
- Syntax:
SELECT column-list FROM table1 NATURAL JOIN table2

Natural Join

FIGURE 0.1
NATURAL JOIN results

SQL> SELECT CUS_CODE, CUS_NAME, CUS_ADDRESS, CUS_CITY, CUS_STATE
2 FROM CUSTOMER WHERE CUS_CITY='NEW YORK';

CUS_CODE	CUS_NAME	CUS_ADDRESS	CUS_CITY
10001	Barney	10001 100-100-100	NEW YORK
10001	Barney	10001 100-100-100	NEW YORK
10001	Barney	10001 100-100-100	NEW YORK
10002	Sally	10002 100-100-100	NEW YORK
10004	Barney	10004 100-100-100	NEW YORK
10004	Barney	10004 100-100-100	NEW YORK
10008	Barney	10008 100-100-100	NEW YORK
10008	Barney	10008 100-100-100	NEW YORK

0 rows returned.

SQL> SELECT ITR_NUMBER, I_CODE, P_DESCRIPTION, LINE_UNITS, LINE_PRICE
2 FROM INVENTORY JOIN ITR WHERE ITR_CITY='NEW YORK';

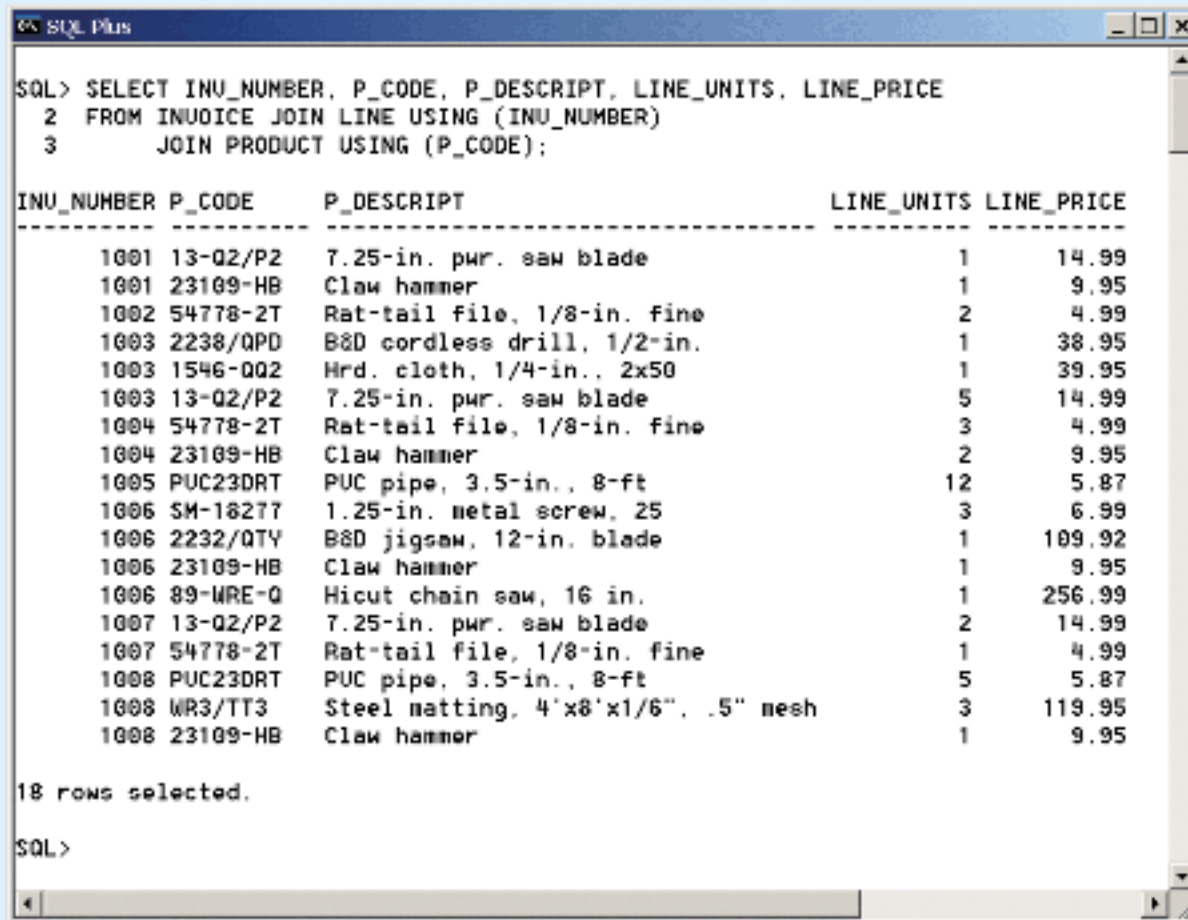
ITR_NUMBER	I_CODE	P_DESCRIPTION	LINE_UNITS	LINE_PRICE
1001 10-100-100	1001	1001 100-100-100	1	19.99
1001 1001-100	1001	1001 100-100-100	1	9.99
1002 10-100-100	1002	1002 100-100-100	2	9.99
1003 1003-100	1003	1003 100-100-100	2	19.99
1004 10-100-100	1004	1004 100-100-100	1	19.99
1005 10-100-100	1005	1005 100-100-100	2	19.99
1006 1006-100	1006	1006 100-100-100	2	9.99
1007 1007-100	1007	1007 100-100-100	2	9.99
1008 1008-100	1008	1008 100-100-100	12	9.99
1009 1009-100	1009	1009 100-100-100	3	9.99
1010 1010-100	1010	1010 100-100-100	1	19.99
1011 1011-100	1011	1011 100-100-100	1	9.99
1012 1012-100	1012	1012 100-100-100	1	9.99
1013 1013-100	1013	1013 100-100-100	1	9.99
1014 1014-100	1014	1014 100-100-100	1	9.99
1015 1015-100	1015	1015 100-100-100	1	9.99
1016 1016-100	1016	1016 100-100-100	1	9.99
1017 1017-100	1017	1017 100-100-100	1	9.99
1018 1018-100	1018	1018 100-100-100	1	9.99
1019 1019-100	1019	1019 100-100-100	1	9.99
1020 1020-100	1020	1020 100-100-100	1	9.99
1021 1021-100	1021	1021 100-100-100	1	9.99
1022 1022-100	1022	1022 100-100-100	1	9.99
1023 1023-100	1023	1023 100-100-100	1	9.99
1024 1024-100	1024	1024 100-100-100	1	9.99
1025 1025-100	1025	1025 100-100-100	1	9.99
1026 1026-100	1026	1026 100-100-100	1	9.99
1027 1027-100	1027	1027 100-100-100	1	9.99
1028 1028-100	1028	1028 100-100-100	1	9.99
1029 1029-100	1029	1029 100-100-100	1	9.99
1030 1030-100	1030	1030 100-100-100	1	9.99
1031 1031-100	1031	1031 100-100-100	1	9.99
1032 1032-100	1032	1032 100-100-100	1	9.99
1033 1033-100	1033	1033 100-100-100	1	9.99
1034 1034-100	1034	1034 100-100-100	1	9.99
1035 1035-100	1035	1035 100-100-100	1	9.99
1036 1036-100	1036	1036 100-100-100	1	9.99
1037 1037-100	1037	1037 100-100-100	1	9.99
1038 1038-100	1038	1038 100-100-100	1	9.99
1039 1039-100	1039	1039 100-100-100	1	9.99
1040 1040-100	1040	1040 100-100-100	1	9.99
1041 1041-100	1041	1041 100-100-100	1	9.99
1042 1042-100	1042	1042 100-100-100	1	9.99
1043 1043-100	1043	1043 100-100-100	1	9.99
1044 1044-100	1044	1044 100-100-100	1	9.99
1045 1045-100	1045	1045 100-100-100	1	9.99
1046 1046-100	1046	1046 100-100-100	1	9.99
1047 1047-100	1047	1047 100-100-100	1	9.99
1048 1048-100	1048	1048 100-100-100	1	9.99
1049 1049-100	1049	1049 100-100-100	1	9.99
1050 1050-100	1050	1050 100-100-100	1	9.99
1051 1051-100	1051	1051 100-100-100	1	9.99
1052 1052-100	1052	1052 100-100-100	1	9.99
1053 1053-100	1053	1053 100-100-100	1	9.99
1054 1054-100	1054	1054 100-100-100	1	9.99
1055 1055-100	1055	1055 100-100-100	1	9.99
1056 1056-100	1056	1056 100-100-100	1	9.99
1057 1057-100	1057	1057 100-100-100	1	9.99
1058 1058-100	1058	1058 100-100-100	1	9.99
1059 1059-100	1059	1059 100-100-100	1	9.99
1060 1060-100	1060	1060 100-100-100	1	9.99
1061 1061-100	1061	1061 100-100-100	1	9.99
1062 1062-100	1062	1062 100-100-100	1	9.99
1063 1063-100	1063	1063 100-100-100	1	9.99
1064 1064-100	1064	1064 100-100-100	1	9.99
1065 1065-100	1065	1065 100-100-100	1	9.99
1066 1066-100	1066	1066 100-100-100	1	9.99
1067 1067-100	1067	1067 100-100-100	1	9.99
1068 1068-100	1068	1068 100-100-100	1	9.99
1069 1069-100	1069	1069 100-100-100	1	9.99
1070 1070-100	1070	1070 100-100-100	1	9.99
1071 1071-100	1071	1071 100-100-100	1	9.99
1072 1072-100	1072	1072 100-100-100	1	9.99
1073 1073-100	1073	1073 100-100-100	1	9.99
1074 1074-100	1074	1074 100-100-100	1	9.99
1075 1075-100	1075	1075 100-100-100	1	9.99
1076 1076-100	1076	1076 100-100-100	1	9.99
1077 1077-100	1077	1077 100-100-100	1	9.99
1078 1078-100	1078	1078 100-100-100	1	9.99
1079 1079-100	1079	1079 100-100-100	1	9.99
1080 1080-100	1080	1080 100-100-100	1	9.99
1081 1081-100	1081	1081 100-100-100	1	9.99
1082 1082-100	1082	1082 100-100-100	1	9.99
1083 1083-100	1083	1083 100-100-100	1	9.99
1084 1084-100	1084	1084 100-100-100	1	9.99
1085 1085-100	1085	1085 100-100-100	1	9.99
1086 1086-100	1086	1086 100-100-100	1	9.99
1087 1087-100	1087	1087 100-100-100	1	9.99
1088 1088-100	1088	1088 100-100-100	1	9.99
1089 1089-100	1089	1089 100-100-100	1	9.99
1090 1090-100	1090	1090 100-100-100	1	9.99
1091 1091-100	1091	1091 100-100-100	1	9.99
1092 1092-100	1092	1092 100-100-100	1	9.99
1093 1093-100	1093	1093 100-100-100	1	9.99
1094 1094-100	1094	1094 100-100-100	1	9.99
1095 1095-100	1095	1095 100-100-100	1	9.99
1096 1096-100	1096	1096 100-100-100	1	9.99
1097 1097-100	1097	1097 100-100-100	1	9.99
1098 1098-100	1098	1098 100-100-100	1	9.99
1099 1099-100	1099	1099 100-100-100	1	9.99
1100 1100-100	1100	1100 100-100-100	1	9.99
1101 1101-100	1101	1101 100-100-100	1	9.99
1102 1102-100	1102	1102 100-100-100	1	9.99
1103 1103-100	1103	1103 100-100-100	1	9.99
1104 1104-100	1104	1104 100-100-100	1	9.99
1105 1105-100	1105	1105 100-100-100	1	9.99
1106 1106-100	1106	1106 100-100-100	1	9.99
1107 1107-100	1107	1107 100-100-100	1	9.99
1108 1108-100	1108	1108 100-100-100	1	9.99
1109 1109-100	1109	1109 100-100-100	1	9.99
1110 1110-100	1110	1110 100-100-100	1	9.99
1111 1111-100	1111	1111 100-100-100	1	9.99
1112 1112-100	1112	1112 100-100-100	1	9.99
1113 1113-100	1113	1113 100-100-100	1	9.99
1114 1114-100	1114	1114 100-100-100	1	9.99
1115 1115-100	1115	1115 100-100-100	1	9.99
1116 1116-100	1116	1116 100-100-100	1	9.99
1117 1117-100	1117	1117 100-100-100	1	9.99
1118 1118-100	1118	1118 100-100-100	1	9.99
1119 1119-100	1119	1119 100-100-100	1	9.99
1120 1120-100	1120	1120 100-100-100	1	9.99
1121 1121-100	1121	1121 100-100-100	1	9.99
1122 1122-100	1122	1122 100-100-100	1	9.99
1123 1123-100	1123	1123 100-100-100	1	9.99
1124 1124-100	1124	1124 100-100-100	1	9.99
1125 1125-100	1125	1125 100-100-100	1	9.99
1126 1126-100	1126	1126 100-100-100	1	9.99
1127 1127-100	1127	1127 100-100-100	1	9.99
1128 1128-100	1128	1128 100-100-100	1	9.99
1129 1129-100	1129	1129 100-100-100	1	9.99
1130 1130-100	1130	1130 100-100-100	1	9.99
1131 1131-100	1131	1131 100-100-100	1	9.99
1132 1132-100	1132	1132 100-100-100	1	9.99
1133 1133-100	1133	1133 100-100-100	1	9.99
1134 1134-100	1134	1134 100-100-100	1	9.99
1135 1135-100	1135	1135 100-100-100	1	9.99
1136 1136-100	1136	1136 100-100-100	1	9.99
1137 1137-100	1137	1137 100-100-100	1	9.99
1138 1138-100	1138	1138 100-100-100	1	9.99
1139 1139-100	1139	1139 100-100-100	1	9.99
1140 1140-100	1140	1140 100-100-100	1	9.99
1141 1141-100	1141	1141 100-100-100	1	9.99
1142 1142-100	1142	1142 100-100-100	1	9.99
1143 1143-100	1143	1143 100-100-100	1	9.99
1144 1144-100	1144	1144 100-100-100	1	9.99
1145 1145-100	1145	1145 100-100-100	1	9.99
1146 1146-100	1146	1146 100-100-100	1	9.99
1147 1147-100	1147	1147 100-100-100	1	9.99
1148 1148-100	1148	1148 100-100-100	1	9.99
1149 1149-100	1149	1149 100-100-100	1	9.99
1150 1150-100	1150	1150 100-100-100	1	9.99
1151 1151-100	1151	1151 100-100-100	1	9.99
1152 1152-100	1152	1152 100-100-100	1	9.99
1153 1153-100	1153	1153 100-100-100	1	9.99
1154 1154-100	1154	1154 100-100-100	1	9.99
1155 1155-100	1155	1155 100-100-100	1	9.99
1156 1156-100	1156	1156 100-100-100	1	9.99
1157 1157-100	1157	1157 100-100-100	1	9.99
1158 1158-100	1158	1158 100-100-100	1	9.99
1159 1159-100	1159	1159 100-100-100	1	9.99
1160 1160-100	1160	1160 100-100-100	1	9.99
1161 1161-100	1161	1161 100-100-100	1	9.99
1162 1162-100	1162	1162 100-100-100	1	9.99
1163 1163-100	1163	1163 100-100-100	1	9.99
1164 1164-100	1164	1164 100-100-100	1	9.99
1165 1165-100	1165	1165 100-100-100	1	9.99
1166 1166-100	1166	1166 100-100-100	1	9.99
1167 1167-100	1167	1167 100-100-100	1	9.99
1168 1168-100	1168	1168 100-100-100	1	9.99
1169 1169-100	1169	1169 100-100-100	1	9.99
1170 1170-100	1170	1170 100-100-100	1	9.99
1171 1171-				

JOIN USING Clause

- Returns only rows with matching values in the column indicated in the USING clause
 - The column must exist in both tables
- Syntax:
`SELECT column-list FROM table1 JOIN table2 USING (common-column)`
- JOIN USING operand does not require table qualifiers
 - Oracle returns error if table name is specified

JOIN USING Clause

FIGURE 8.2 JOIN USING results



The screenshot shows an SQL Plus window with a query and its results. The query is a JOIN USING statement that joins the INVOICE, LINE, and PRODUCT tables. The results are displayed in a table with columns INU_NUMBER, P_CODE, P_DESCRIPTOR, LINE_UNITS, and LINE_PRICE. There are 18 rows of data.

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

INU_NUMBER	P_CODE	P_DESCRIPTOR	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	PUC23DRT	PUC pipe, 3.5-in., 8-ft	12	5.87
1006	SM-18277	1.25-in. metal screw, 25	3	6.99
1006	2232/QT4	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	PUC23DRT	PUC 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>
```

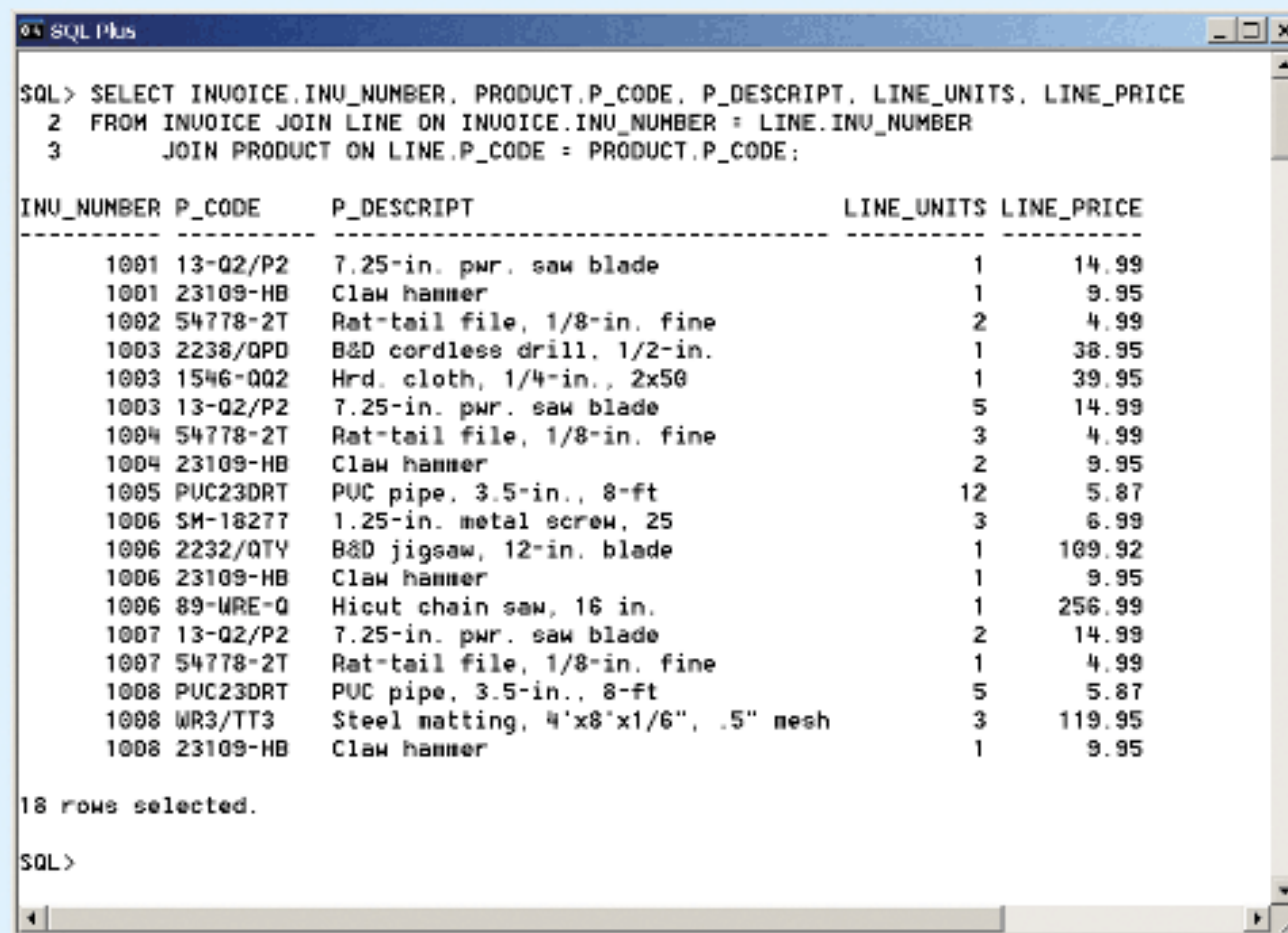
SOURCE: Course Technology/Cengage Learning

JOIN ON Clause

- Used when tables have no common attributes
- Returns only rows that meet the join condition
 - Typically includes equality comparison expression of two columns
 - Column names need not be the same but they must be the same data type
- Syntax:
`SELECT column-list FROM table1 JOIN table2 ON join-condition`

JOIN ON Clause

FIGURE 8.3 JOIN ON results



The screenshot shows an SQL Plus window with a query that joins the INVOICE, LINE, and PRODUCT tables. The query uses the JOIN ON clause to specify the join conditions. The result is a table with 18 rows, showing invoice numbers, product codes, descriptions, units, and prices.

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

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	PUC23DRT	PUC pipe, 3.5-in., 8-ft	12	5.87
1006	SM-18277	1.25-in. metal screw, 25	3	6.99
1006	2232/QT4	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	PUC23DRT	PUC 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>

SOURCE: Course Technology/Cengage Learning

JOIN ON Clause

- The following query will generate a list of all employees with the managers' names

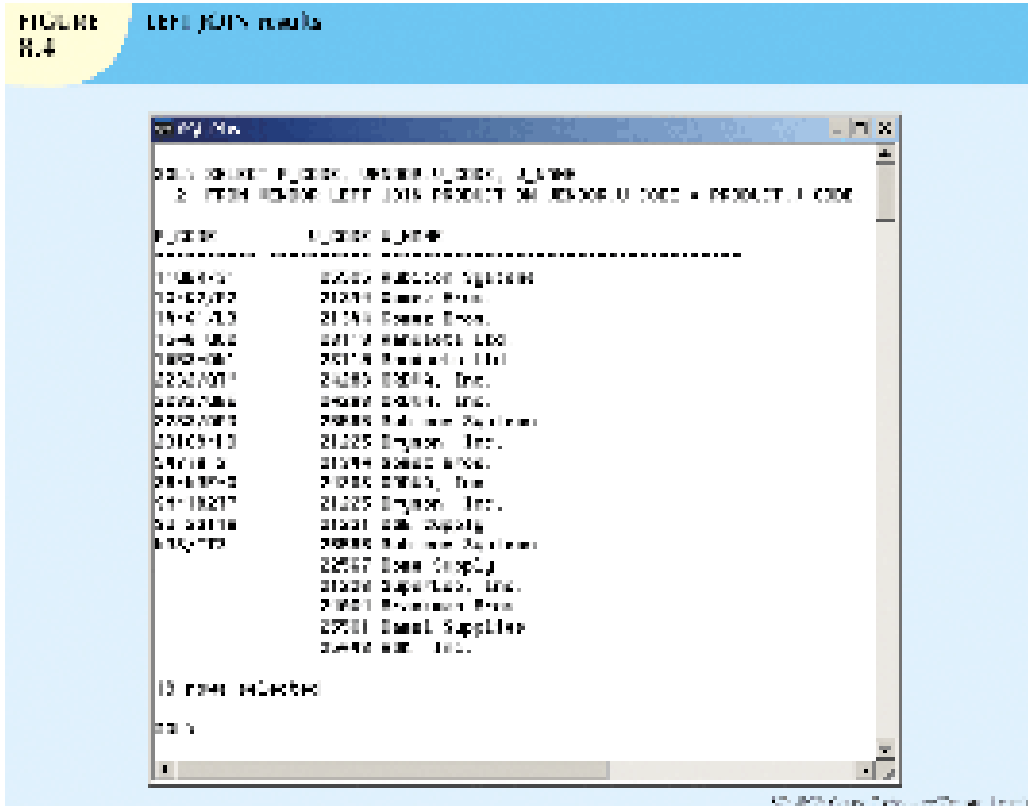
```
SELECT E.EMP_MGR, M.EMP_LNAME, E.EMP_NUM, E.EMP_LNAME  
FROM   EMP E JOIN EMP M ON E.EMP_MGR = M.EMP_NUM  
ORDER BY E.EMP_MGR
```

Outer Joins

- Returns rows matching the join condition
- Also returns rows with unmatched attribute values for tables to be joined
- Three types
 - Left
 - Right
 - Full
- Left and right designate order in which tables are processed

Outer Joins (cont'd.)

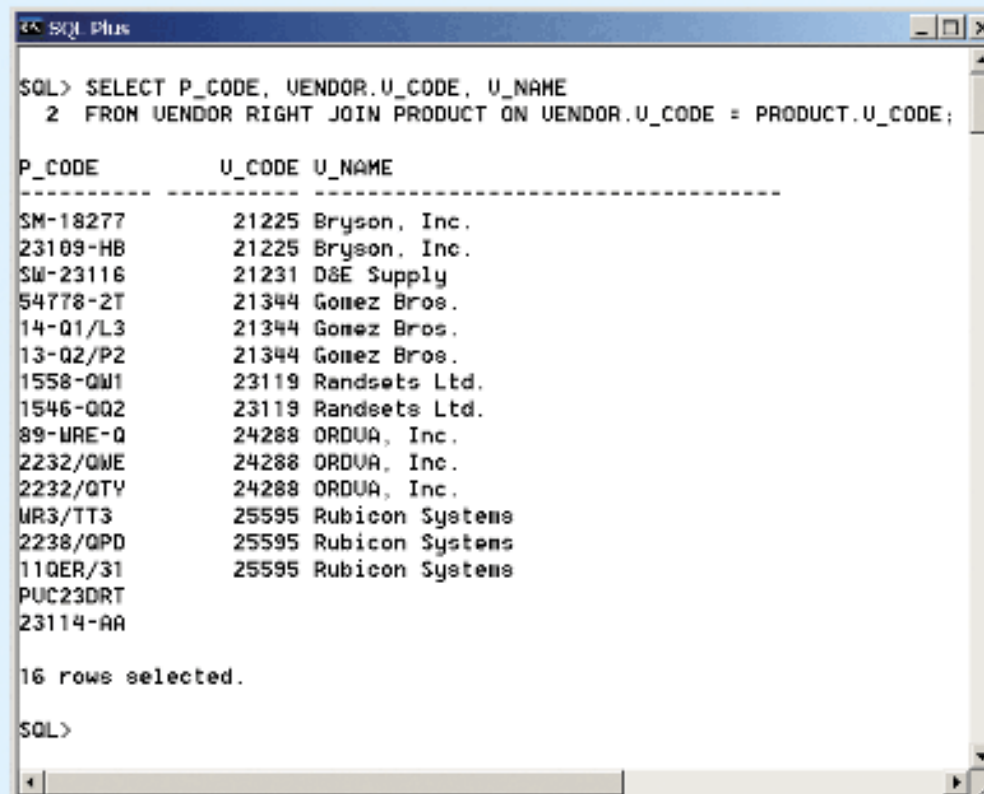
- Left outer join
 - Returns rows matching the join condition
 - Returns rows in left side table with unmatched values
 - Syntax: SELECT column-list FROM table1 LEFT [OUTER] JOIN table2 ON join-condition



Outer Joins (cont'd.)

- Right outer join
 - Returns rows matching join condition
 - Returns rows in right side table with unmatched values

FIGURE 8.5 RIGHT JOIN results



The screenshot shows an SQL Plus window with the following content:

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

P_CODE	U_CODE	U_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	Randssets Ltd.
1546-QQ2	23119	Randssets Ltd.
89-WRE-Q	24288	ORDUA, Inc.
2232/QWE	24288	ORDUA, Inc.
2232/QTU	24288	ORDUA, Inc.
MR3/TT3	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
11QER/31	25595	Rubicon Systems
PUC23DRT		
23114-AA		

16 rows selected.

SQL>

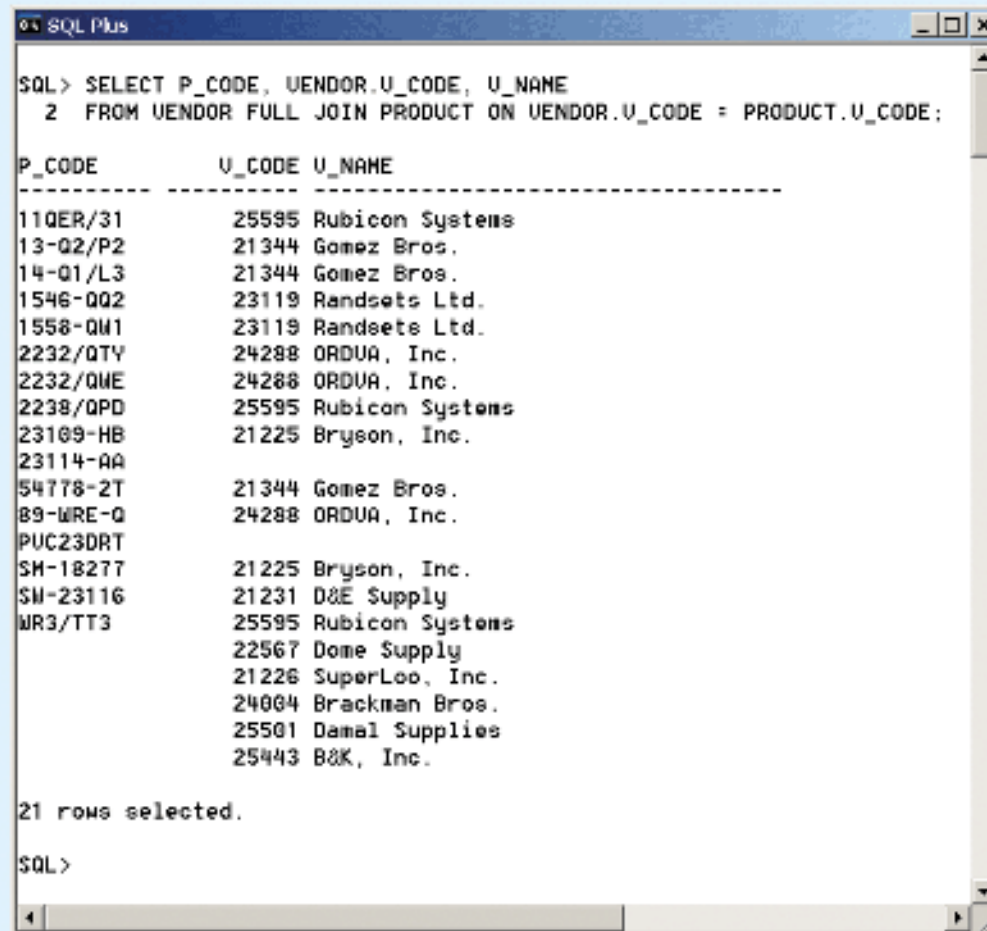
SOURCE: Course Technology/Cengage Learning

Outer Joins (cont'd.)

- Full outer join
 - Returns rows matching join condition
 - Returns all rows with unmatched values in either side table
 - Syntax:

```
SELECT column-list  
FROM table1 FULL [OUTER] JOIN table2  
ON join-condition
```

FIGURE 8.6 FULL JOIN results



The screenshot shows an SQL Plus window with the following content:

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

P_CODE	U_CODE	U_NAME
11QER/31	25595	Rubicon Systems
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
1546-QQ2	23119	Randsets Ltd.
1558-QM1	23119	Randsets Ltd.
2232/QTU	24288	ORDUA, Inc.
2232/QME	24288	ORDUA, Inc.
2238/QPD	25595	Rubicon Systems
23109-HB	21225	Bryson, Inc.
23114-AA		
54778-2T	21344	Gomez Bros.
89-WRE-Q	24288	ORDUA, Inc.
PUC23DRT		
SH-18277	21225	Bryson, Inc.
SH-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.

21 rows selected.

```
SQL>
```

SOURCE: Course Technology/Cengage Learning

Relational Set Operators

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

UNION

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

- Example:

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

- Can be used to unite more than two queries

UNION

**FIGURE
8.16**

UNION query results

Database name: CH08_SaleCo

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
10010	Ramos	Alice	A	815	844-2573	0.00
10011	Dunne	Leona	K	713	894-1238	0.00
10012	Smith	Kathy	W	815	894-2389	945.86
10013	Olowoski	Paul	F	815	894-2180	698.75
10014	Orlando	Myron		815	222-1872	0.00
10015	O'Brien	Amy	B	713	442-9981	0.00
10016	Ernst	James	O	815	287-1238	221.19
10017	Williams	George		815	280-2596	768.93
10018	Farriss	Anne	O	713	382-7185	216.55
10019	Smith	Orla	K	815	287-3809	0.00

Table name: CUSTOMER_2

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
345	Tennel	Justine	H	815	322-8070
347	Olowoski	Paul	F	815	894-2180
351	Hernandez	Carlos	J	723	123-7804
352	McDowell	George		723	123-7768
355	Trinh	Khaleed	G	723	123-8875
359	Lewis	Maria	J	734	332-1789
363	Dunne	Leona	K	713	894-1238

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

CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
Brown	James	O	815	287-1238
Dunne	Leona	K	713	894-1238
Farriss	Anne	O	713	382-7185
Hernandez	Carlos	J	723	123-7804
Lewis	Maria	J	734	332-1789
McDowell	George		723	123-7768
O'Brien	Amy	B	713	442-9981
Olowoski	Paul	F	815	894-2180
Orlando	Myron		815	222-1872
Ramos	Alice	A	815	844-2573
Smith	Kathy	W	815	894-2389
Smith	Orla	K	815	287-3809
Tennel	Justine	H	815	322-8070
Trinh	Khaleed	G	723	123-8875
Williams	George		815	280-2596

SOURCE: Course Technology/Cengage Learning

UNION ALL

- Produces a relation that retains duplicate rows

- Example query:

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

- Can be used to unite more than two queries

UNION ALL

FIGURE 8.17

UNION ALL query results

Database name: CH08_SaleCo

Table name: CUSTOMER

Query name: qryUNION-ALL-of-CUSTOMER-and-CUSTOMER_2

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
10010	Ramoa	Alfred	A	615	884-2573	0.00
10011	Dunna	Leona	K	713	884-1238	0.00
10012	Smith	Kathy	W	615	884-2585	345.55
10013	Olowodi	Paul	F	615	884-2180	535.75
10014	Orlando	Myron		615	223-1672	0.00
10015	O'Brien	Amy	B	713	443-3381	0.00
10016	Brown	James	G	615	287-1228	221.18
10017	Williams	George		615	280-2226	760.80
10018	Parlier	Anne	G	713	383-7185	316.55
10019	Smith	Olivia	K	615	287-3009	0.00

Table name: CUSTOMER_2

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
345	Tennil	Justine	H	615	322-8870
347	Olowodi	Paul	F	615	884-2180
351	Hernandez	Carlos	J	723	123-7854
353	McDowell	George		723	123-7765
365	Tipin	Khalid	G	723	123-8875
368	Lewis	Marie	J	734	333-1709
369	Dunna	Leona	K	713	884-1238

CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
Ramoa	Alfred	A	615	884-2573
Dunna	Leona	K	713	884-1238
Smith	Kathy	W	615	884-2585
Olowodi	Paul	F	615	884-2180
Orlando	Myron		615	223-1672
O'Brien	Amy	B	713	443-3381
Brown	James	G	615	287-1228
Williams	George		615	280-2226
Parlier	Anne	G	713	383-7185
Smith	Olivia	K	615	287-3009
Tennil	Justine	H	615	322-8870
Olowodi	Paul	F	615	884-2180
Hernandez	Carlos	J	723	123-7854
McDowell	George		723	123-7765
Tipin	Khalid	G	723	123-8875
Lewis	Marie	J	734	333-1709
Dunna	Leona	K	713	884-1238

SOURCE: Course Technology/Cengage Learning

INTERSECT

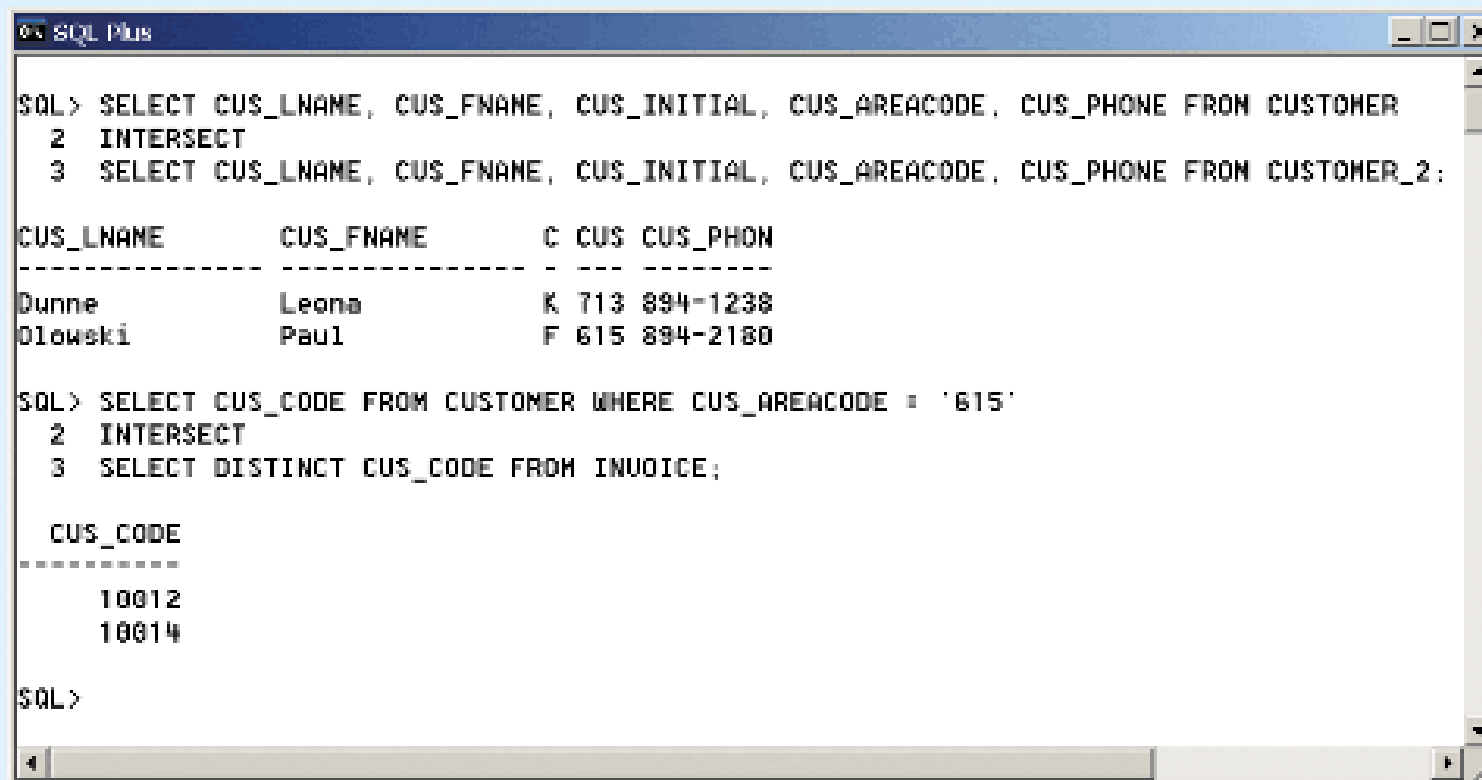
- Combines rows from two queries, returning only the rows that appear in both sets
- Syntax: query INTERSECT query

- Example query:

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

**FIGURE
8.18**

INTERSECT query results



```
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>
```

SOURCE: Course Technology/Cengage Learning

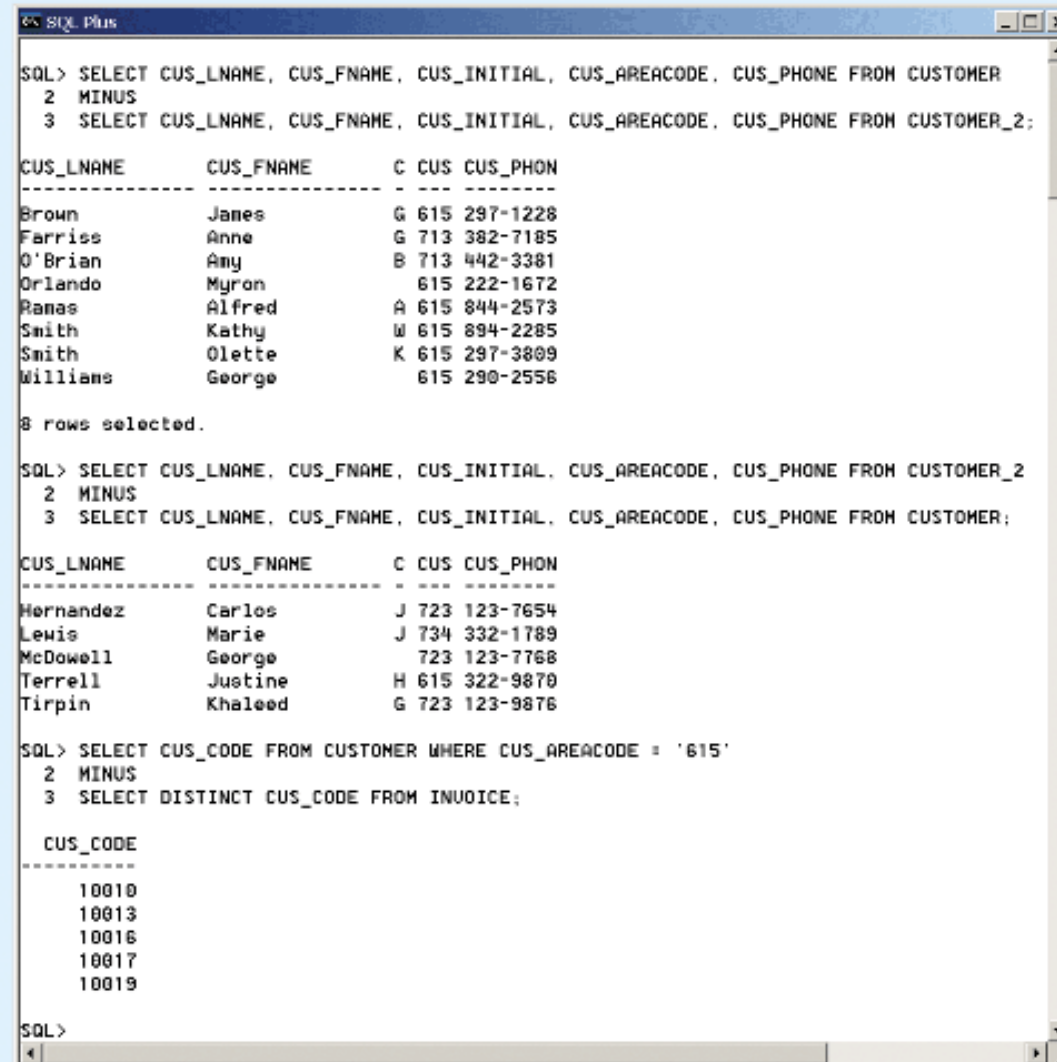
MINUS

- Combines rows from two queries
 - Returns only the rows that appear in the first set but not in the second
- Syntax: query MINUS query (MS SQL uses EXCEPT)
 - Example:

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


MINUS

FIGURE 8.19 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	Janes	G	615	297-1228
Farriss	Anne	G	713	382-7185
O'Brian	Any	B	713	442-3381
Orlando	Myron		615	222-1672
Ranas	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>
```

SOURCE: Course Technology/Cengage Learning

Syntax Alternatives

- IN and NOT IN subqueries can be used in place of INTERSECT or MINUS

Example: INTERSECT

```
SELECT  CUS_CODE FROM CUSTOMER
```

```
WHERE   CUS_AREACODE = '615' AND
```

```
        CUS_CODE IN
```

```
        (SELECT DISTINCT CUS_CODE FROM INVOICE);
```

- MINUS would use NOT IN

Subqueries and Correlated Queries

- Often necessary to process data based on other processed data
- Subquery is a query inside a query, normally inside parentheses
- First query is the outer query
 - Inside query is the inner query
- Inner query is executed first
- Output of inner query is used as input for outer query
- Sometimes referred to as a nested query

Subqueries and Correlated Queries

TABLE 8.2 SELECT Subquery Examples

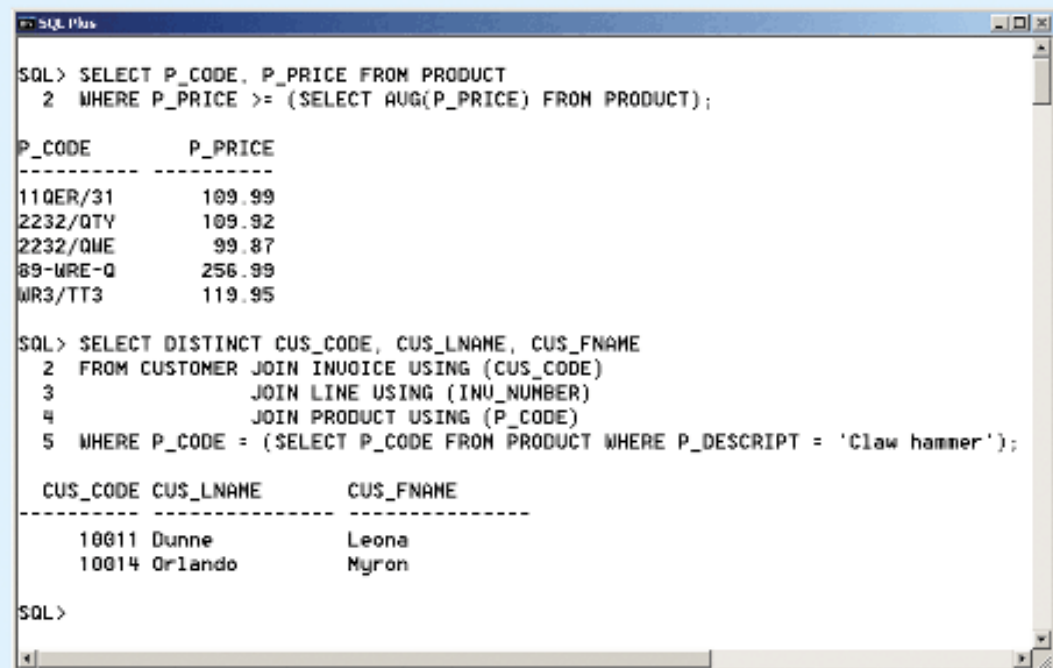
SELECT SUBQUERY EXAMPLES	EXPLANATION
INSERT INTO PRODUCT SELECT * FROM P;	Inserts all rows from Table P into the PRODUCT table. Both tables must have the same attributes. The subquery returns all rows from Table P.
UPDATE PRODUCT SET P_PRICE = (SELECT AVG(P_PRICE) FROM PRODUCT) WHERE V_CODE IN (SELECT V_CODE FROM VENDOR WHERE V_AREACODE = '615')	Updates the product price to the average product price, but only for products 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 provided by vendors with an area code equal to 615. The subquery returns the list of vendor codes with an area code equal to 615.

WHERE Subqueries

- Most common type uses inner SELECT subquery on right side of WHERE comparison
 - Requires a subquery that returns only one single value
- Value generated by subquery must be of comparable data type
- Can be used in combination with joins

FIGURE
8.7

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/DTY      109.92
2232/QME       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 (INU_NUMBER)
 4              JOIN PRODUCT USING (P_CODE)
 5  WHERE P_CODE = (SELECT P_CODE FROM PRODUCT WHERE P_DESCRIPTION = 'Claw hammer');

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

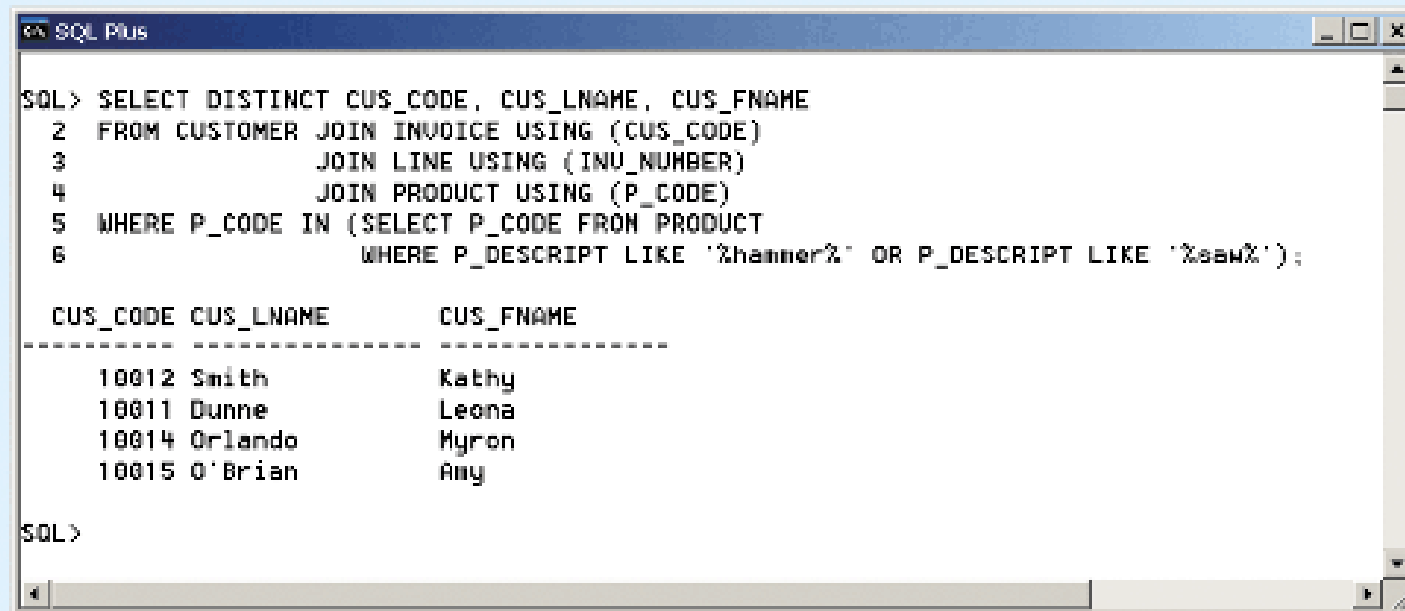
SQL>
```

SOURCE: Course Technology/Cengage Learning

IN Subqueries

- Used when comparing a single attribute to a list of values

FIGURE 8.8 IN subquery example



```
SQL> SELECT DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
2  FROM CUSTOMER JOIN INVOICE USING (CUS_CODE)
3           JOIN LINE USING (INU_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
-----
10012 Smith            Kathy
10011 Dunne            Leona
10014 Orlando          Myron
10015 O'Brian          Amy

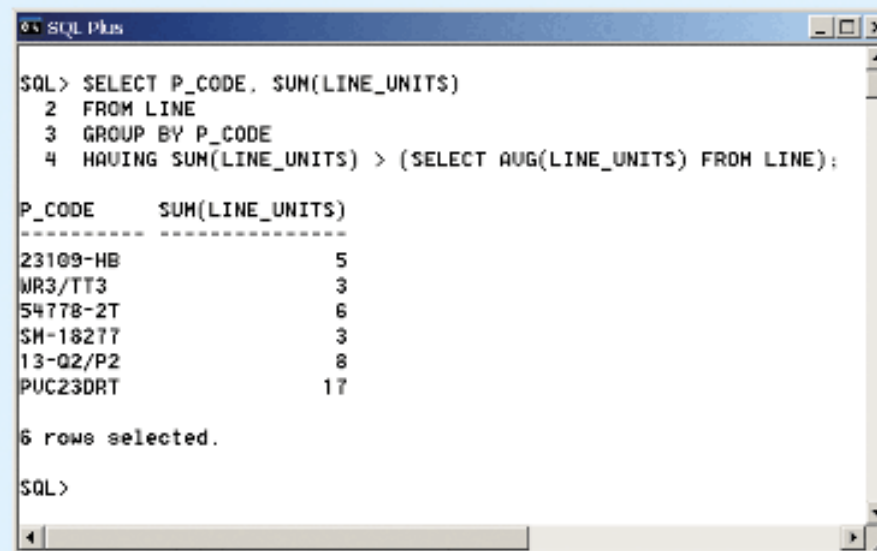
SQL>
```

SOURCE: Course Technology/Cengage Learning

HAVING Subqueries

- HAVING clause restricts the output of a GROUP BY query
 - Applies conditional criterion to the grouped rows

FIGURE 8.9 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)
-----
23109-HB          5
WR3/TT3           3
54778-2T           6
SM-18277           3
13-Q2/P2           8
PUC23DRT          17

6 rows selected.

SQL>
```

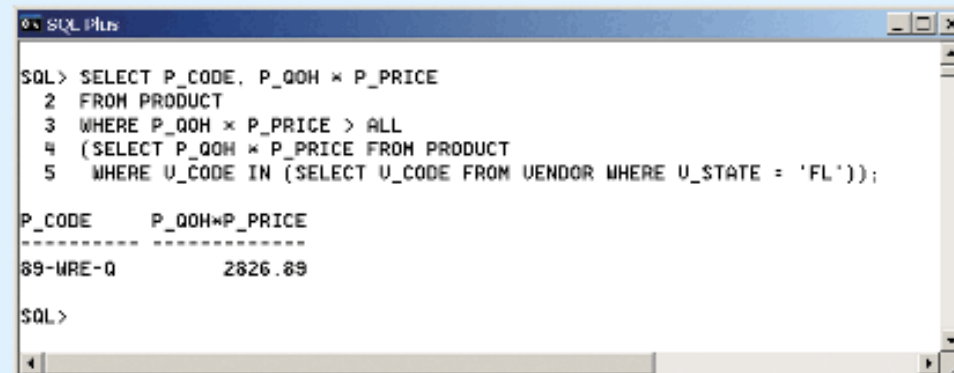
SOURCE: Course Technology/Cengage Learning

Multirow Subquery Operators: ANY and ALL

- Allows comparison of single value with a list of values using inequality comparison
 - “Greater than ALL” equivalent to “greater than the highest in list”
 - “Less than ALL” equivalent to “less than lowest”
 - Using equal to ANY operator equivalent to IN operator
 - In the query below we compare a single value ($P_QOH \times P_PRICE$) with a list of values returned by the first subquery
 - For a row to appear in the resultset, it has to meet the criterion $P_QOH \times P_PRICE > \text{ALL}$ of the individual values returned by the first subquery

FIGURE 8.10

Multirow subquery operator example



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

P_CODE      P_QOH*P_PRICE
-----
89-MRE-Q          2826.89

SQL>
```

SOURCE: Course Technology/Cengage Learning


```
SELECT V_CODE FROM VENDOR WHERE V_STATE='FL';
```

V_CODE

21226

25443

25595

```
SELECT P_QOH*P_PRICE FROM PRODUCT
```

```
WHERE V_CODE IN (SELECT V_CODE FROM VENDOR WHERE V_STATE='FL');
```

QOH*PRICE

879.92

467.4

2159.1

```
SELECT P_CODE, P_QOH*P_PRICE, V_STATE
```

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

```
WHERE P_QOH*P_PRICE > ALL
```

```
(SELECT P_QOH*P_PRICE FROM PRODUCT
```

```
WHERE V_CODE IN (SELECT V_CODE FROM VENDOR WHERE V_STATE='FL'));
```

P_CODE QOH*PRICE V_STATE

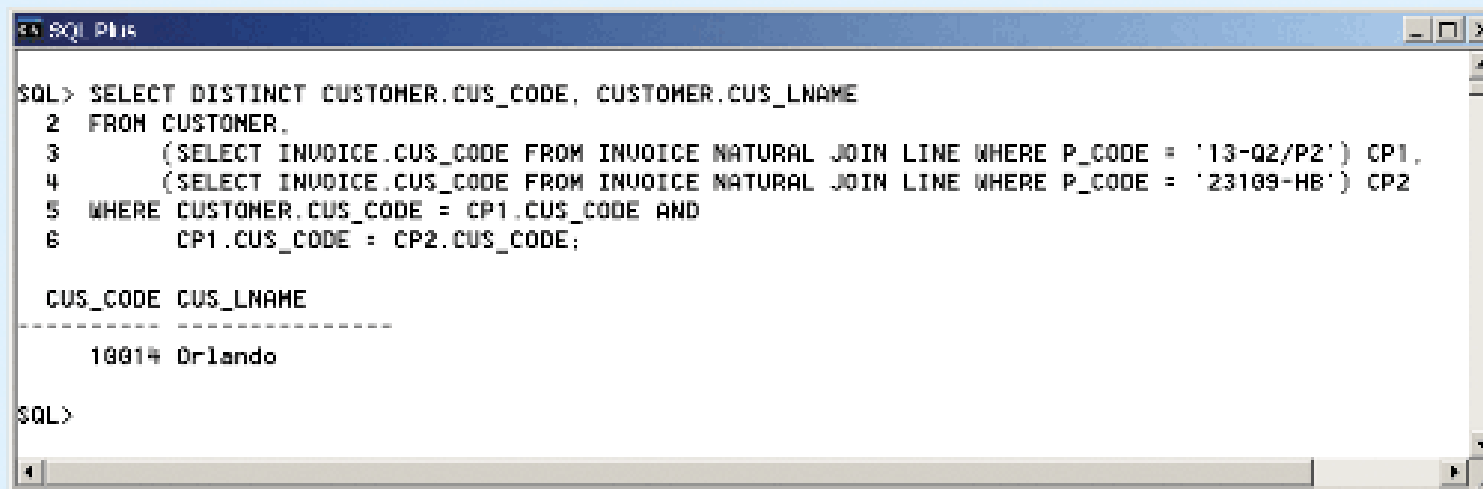
89-WRE-Q2826.89 TN

FROM Subqueries

- Specifies the tables from which the data will be drawn
- Can use SELECT subquery in the FROM clause
 - View name can be used anywhere a table is expected

FIGURE
8.11

FROM subquery example



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

CUS_CODE CUS_LNAME
-----
10014 Orlando

SQL>
```

SOURCE: Course Technology/Cengage Learning

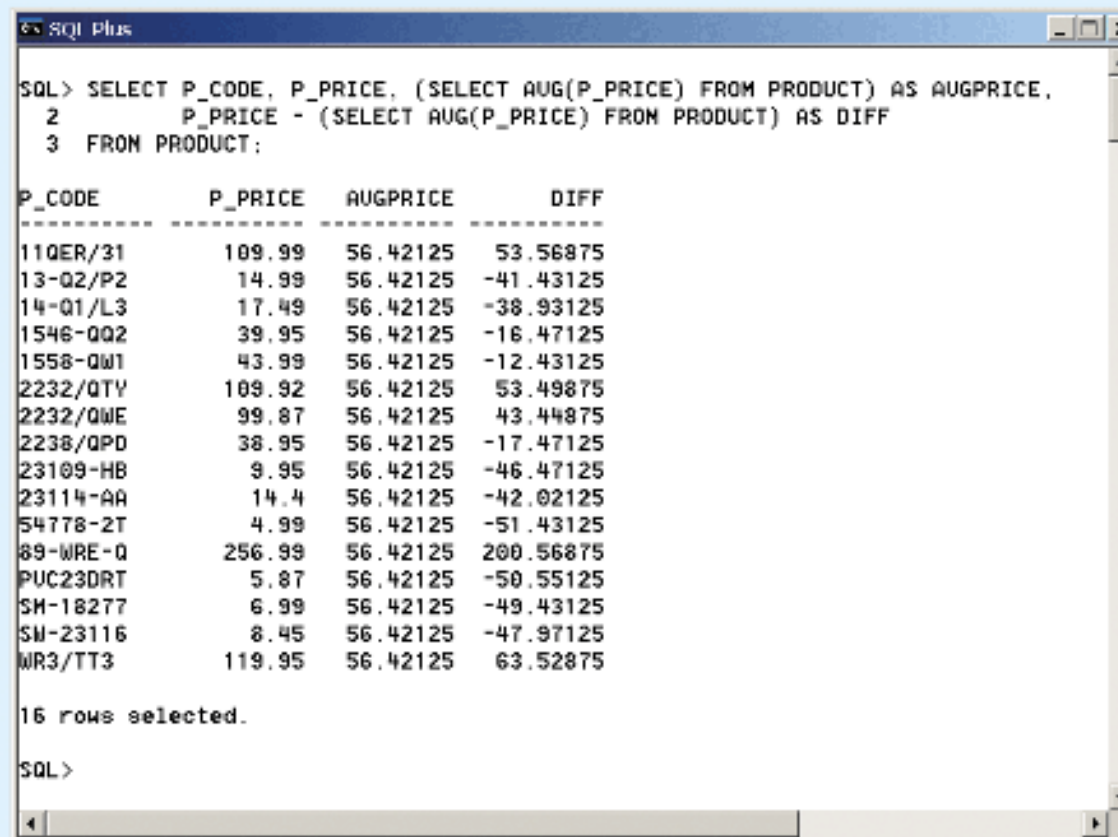
Attribute List Subqueries

- SELECT statement uses attribute list to indicate columns to project resulting set
 - Columns can be attributes of base tables
 - Result of aggregate function
- Attribute list can also include subquery expression: inline subquery
 - Must return one single value

Attribute List Subqueries

- Cannot use an alias in the attribute list
 - You can not write P_PRICE-AVGPRICE in line 2

FIGURE 8.12 Inline subquery example



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

P_CODE	P_PRICE	AUGPRICE	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
PUC23DRT	5.87	56.42125	-50.55125
SH-18277	6.99	56.42125	-49.43125
SM-23116	8.45	56.42125	-47.97125
WR3/TT3	119.95	56.42125	63.52875

16 rows selected.

SQL>

SOURCE: Course Technology/Cengage Learning

Correlated Subqueries

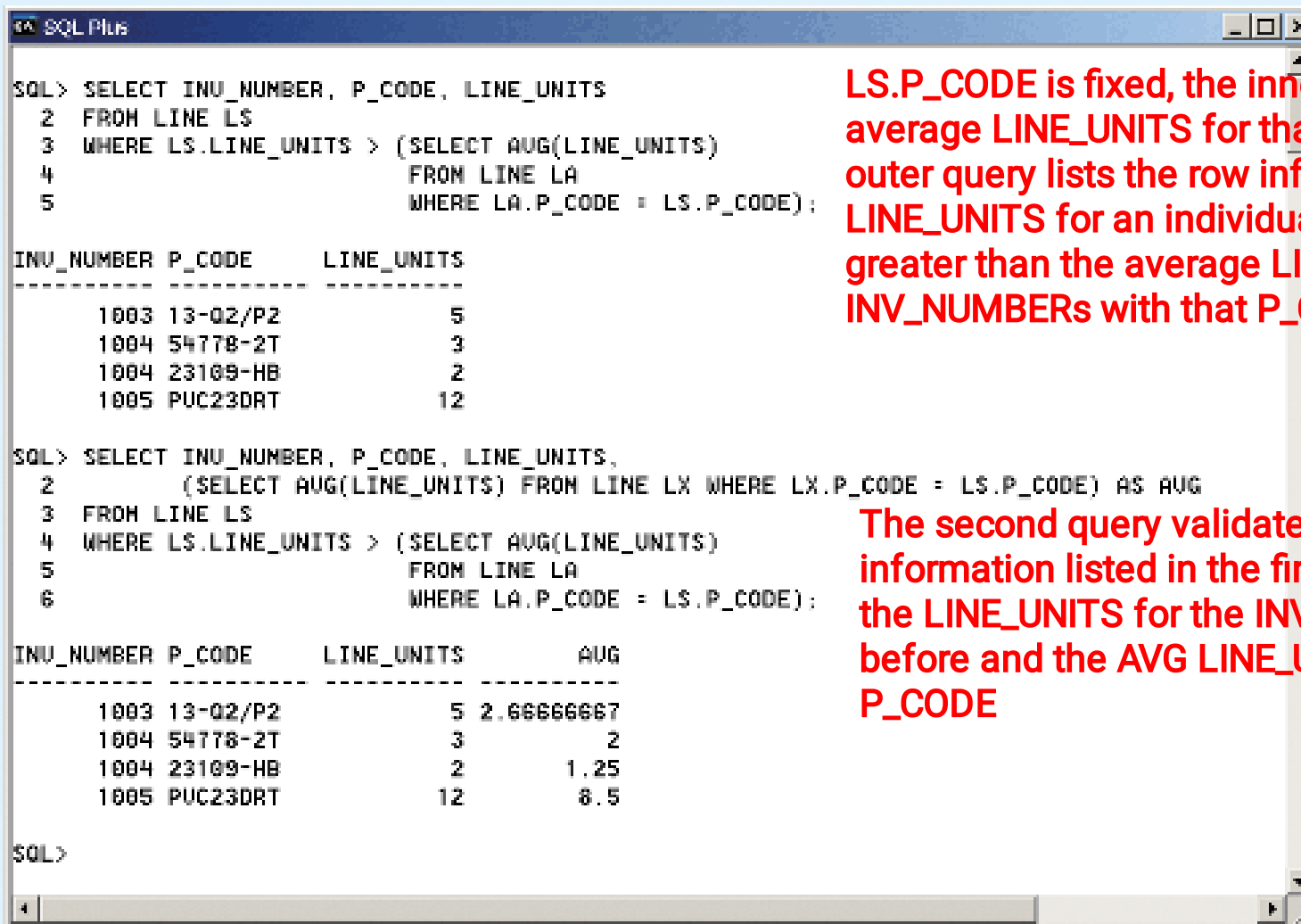
- Subquery that executes once for each row in the outer query
- Correlated because inner query is related to the outer query
 - Inner query references column of outer subquery
 - Works like a nested loop

```
for x=1 to 2
  for y=1 to 3
    print "x=" x "y=" y
```

as x holds steady with 1, y loops from 1 to 3
- Can also be used with the EXISTS special operator

Correlated Subqueries

FIGURE 8.14 Correlated subquery examples



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

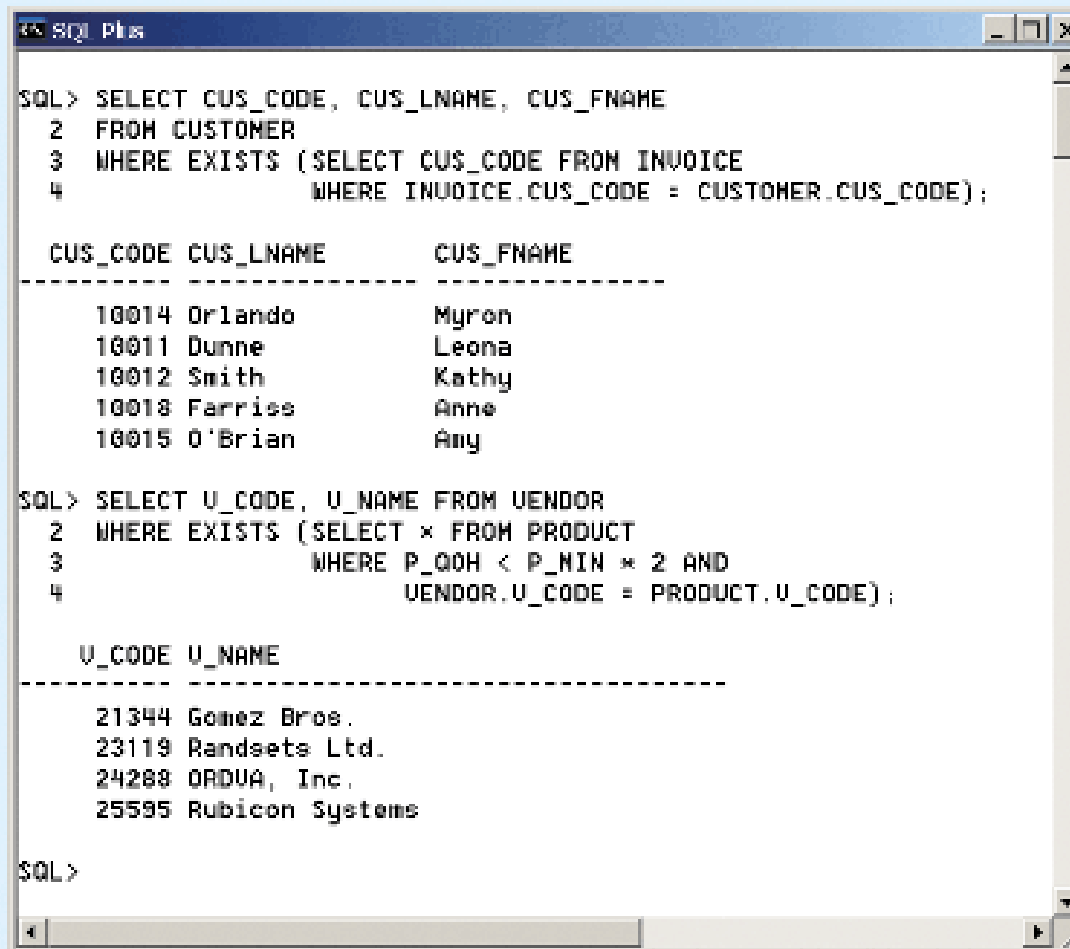
LS.P_CODE is fixed, the inner query finds the average LINE_UNITS for that code and the outer query lists the row information if the LINE_UNITS for an individual INV_NUMBER is greater than the average LINE_UNITS for all INV_NUMBERS with that P_CODE

The second query validates the information listed in the first by showing the LINE_UNITS for the INV_NUMBER as before and the AVG LINE_UNITS for that P_CODE

EXISTS Correlated Subquery

FIGURE
8.15

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

SQL> SELECT U_CODE, U_NAME FROM VENDOR
2 WHERE EXISTS (SELECT * FROM PRODUCT
3              WHERE P_QOH < P_MIN * 2 AND
4              VENDOR.U_CODE = PRODUCT.U_CODE);

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

SQL>
```

The first query shows the names of all customers who have at least one order in the INVOICE table

The second query shows the vendor info for those products who QOH is less than twice the minimum quantity

SOURCE: Course Technology/Cengage Learning

SQL Functions

- Generating information from data often requires many data manipulations
- SQL functions are similar to functions in programming languages
- Functions always use numerical, date, or string value
- Value may be part of a command or attribute in a table
- Function may appear anywhere in an SQL statement

Date and Time Functions

- All SQL-standard DBMSs support date and time functions
- Date functions take one parameter and return a value
- Date/time data types are implemented differently by different DBMS vendors
- ANSI SQL standard defines date data types, but not how data types are stored

**TABLE
8.3**

Selected MS Access and SQL Server Date/Time Functions

FUNCTION	EXAMPLE(S)
YEAR Returns a four-digit year Syntax: YEAR(date_value)	Lists all employees born in 1966: 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)	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() MS Access GETDATE() SQL Server Returns today's date	Lists how many days are left until Christmas: SELECT #25-Dec-2012# - 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 number signs (#) because you are doing date arithmetic. In MS SQL Server: Use GETDATE() to get the current system date. To compute the difference between dates, use the DATEDIFF function (see below).
DATEADD SQL Server Adds a number of selected time periods to a date Syntax: DATEADD(datepart, number, date)	Adds a number of dateparts to a given date. Dateparts can be minutes, hours, days, weeks, months, quarters, or years. For example: SELECT DATEADD(day,90, P_INDATE) AS DueDate FROM PRODUCT; The preceding example adds 90 days to P_INDATE. In MS Access, use the following: SELECT P_INDATE+90 AS DueDate FROM PRODUCT;
DATEDIFF SQL Server Subtracts two dates Syntax: DATEDIFF(datepart, startdate, enddate)	Returns the difference between two dates expressed in a selected datepart. For example: SELECT DATEDIFF(day, P_INDATE, GETDATE()) AS DaysAgo FROM PRODUCT; In MS Access, use the following: SELECT DATE() - P_INDATE AS DaysAgo FROM PRODUCT;

Numeric Functions

- Grouped in different ways
 - Algebraic, trigonometric, logarithmic, etc.
- Do not confuse with aggregate functions
 - Aggregate functions operate over sets
 - Numeric functions operate over single row
- Numeric functions take one numeric parameter and return one value

**TABLE
8.5**

Selected Numeric Functions

FUNCTION	EXAMPLE(S)
ABS Returns the absolute value of a number Syntax: ABS(numeric_value)	In Oracle, use the following: SELECT 1.95, -1.93, ABS(1.95), ABS(-1.93) FROM DUAL; In MS Access and SQL Server, use the following: SELECT 1.95, -1.93, ABS(1.95), ABS(-1.93);
ROUND Rounds a value to a specified precision (number of digits) Syntax: ROUND(numeric_value, p) p = precision	Lists the product prices rounded to one and zero decimal places: SELECT P_CODE, P_PRICE, ROUND(P_PRICE,1) AS PRICE1, ROUND(P_PRICE,0) AS PRICE0 FROM PRODUCT;
CEIL/CEILING/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) Oracle CEILING(numeric_value) SQL Server FLOOR(numeric_value)	Lists the product price, the smallest integer greater than or equal to the product price, and the largest integer equal to or less than the product price. In Oracle, use the following: SELECT P_PRICE, CEIL(P_PRICE), FLOOR(P_PRICE) FROM PRODUCT; In SQL Server, use the following: SELECT P_PRICE, CEILING(P_PRICE), FLOOR(P_PRICE) FROM PRODUCT; MS Access does not support these functions.

String Functions

- String manipulations are the most used functions in programming
- String manipulation function examples:
 - Concatenation
 - Printing in uppercase
 - Finding length of an attribute

Conversion Functions

- Take a value of given data type and convert it to the equivalent value in another data type
- Oracle conversion functions:
 - TO_CHAR: takes a date value, converts to character string
 - TO_DATE: takes character string representing a date, converts it to actual date in Oracle format

SQL Conversion Functions

- SQL Server uses CAST and CONVERT functions
 - Numeric to Character
 - CAST(numeric as varchar(length))
 - CONVERT(varchar(length), numeric)
 - Date to Character
 - CAST(date as varchar(length))
 - CONVERT(varchar(length), date)
 - String to Number
 - CAST('-123.99' AS NUMERIC(8,2))
 - String to Date
 - CONVERT(DATETIME,'01/JAN/1970')
 - CAST('01-JAN-1970' AS DATETIME)

Virtual Tables: Creating a View

- View
 - Virtual table based on a SELECT query
- Base tables
- Tables on which the view is based
- CREATE VIEW viewname AS SELECT query

```
create view v_empname as  
select emp_lname, emp_fname, emp_initial  
from emp;
```


Aggregate Functions

**TABLE
7.8**

**Some Basic SQL Aggregate
Functions**

FUNCTION	OUTPUT
COUNT	The number of rows containing non-null values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column