Lab 3.1. Basic SELECT Queries

Each clause in a SELECT query performs a specific function. Understanding the function of each clause is key to developing the skills to construct queries to satisfy the reporting needs of the users. The following clauses will be covered in this module. (although not in this order).

• SELECT—specifies the attributes to be returned by the query

• FROM—specifies the table(s) from which the data will be retrieved

• WHERE—filters the rows of data based on provided criteria

• GROUP BY—groups the rows of data into collections based on sharing the same values in one or more attributes

• HAVING—filters the groups formed in the GROUP BY clause based on provided criteria

• ORDER BY—sorts the final query result rows in ascending or descending order based on the values of one or more attributes.

Although SQL commands can be grouped together on a single line, complex command sequences are best shown on separate lines, with space between the SQL command and the command's components. Using that formatting convention makes it much easier to see the components of the SQL statements, which in turn makes it easy to trace the SQL logic and make corrections if necessary. The number of spaces used in the indention is up to you. For a SELECT query to retrieve data from the database, it will require at least a SELECT column list and a FROM clause. The SELECT column list specifies the relational projection, as discussed in The Relational Database Model. The column list allows the programmer to specify which columns should be retrieved by the query and the order in which they should be returned. Only columns specified in the column list will appear in the query result. The FROM clause is used to specify the table from which the data will be retrieved. It is common for queries to retrieve data from multiple tables that have been joined together, as discussed in The Relational Database Model. However, first, we will focus on things that can be done with the column list before we move on to the FROM clause options.

Lab 3.2. SELECT Statement Options

The SELECT query specifies the columns to be retrieved as a column list. The syntax for a basic SELECT query that retrieves data from a table is:

SELECT columnlist

FROM tablelist;

The columnlist represents one or more attributes, separated by commas. If the programmer wants all of the columns to be returned, then the asterisk (*) wildcard can be used. A wildcard character is a symbol that can be used as a general substitute for other characters or commands. This wildcard means "all columns." For example, the following query would return all of the data from the PRODUCT table (see Figure 7.2).

FIGURE 7.2 SELECT AN ENTIRE TABLE

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

In this query, the column list indicates that all columns (and by default all of the rows) should be returned. The FROM clause specifies that the data from the PRODUCT table is to be used. Recall from The Relational Database Model that projection does not limit the rows being returned. To limit the rows being returned, relational selection (or restriction) must be used. The column list allows the programmer to specify which columns should be returned, as shown in the next query (see Figure 7.3).

SELECT P_CODE, P_DESCRIPT, P_PRICE, P_QOH

FROM PRODUCT;

FIGURE 7.3 SELECT WITH A COLUMN LIST

P_CODE	P_DESCRIPT	P_PRICE	P_QOH
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	8
13-Q2/P2	7.25-in. pwr. saw blade	14.99	32
14-Q1/L3	9.00-in. pwr. saw blade	17.49	18
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	15
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23
2232/QTY	B&D jigsaw, 12-in. blade	109.92	8
2232/QWE	B&D jigsaw, 8-in. blade	99.87	6
2238/QPD	B&D cordless drill, 1/2-in.	38.95	12
23109-HB	Claw hammer	9.95	23
23114-AA	Sledge hammer, 12 lb.	14.40	8
54778-2T	Rat-tail file, 1/8-in. fine	4.99	43
89-WRE-Q	Hicut chain saw, 16 in.	256.99	11
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	188
SM-18277	1.25-in. metal screw, 25	6.99	172
SW-23116	2.5-in. wd. screw, 50	8.45	237
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	119.95	18

This query specifies that the data should come from the PRODUCT table, and that only the product code, description, price, and quantity on hand columns should be included. Notice that only the requested columns are returned and that the columns are in the same order in the output as they were listed in the query. To display the columns in a different order, simply change the order of the columns in the column list.

Lab 3.3. FROM Clause Options

The FROM clause of the query specifies the table or tables from which the data is to be retrieved. In the following query, the data is being retrieved from only the PRODUCT table.

```
SELECT P_CODE, P_DESCRIPT, P_INDATE, P_QOH, P_MIN, P_PRICE,
P_DISCOUNT, V_CODE
FROM PRODUCT;
```

Lab 3.4. ORDER BY Clause Options

The ORDER BY clause is especially useful when the listing order is important to you. The syntax is:

```
SELECT columnlist
FROM tablelist
[ORDER BY columnlist [ASC | DESC] ];
```

Although you have the option of declaring the order type—ascending or descending— the default order is ascending. For example, if you want the contents of the PRODUCT table to be listed by P_PRICE in ascending order, use the following command:

```
SELECT P_CODE, P_DESCRIPT, P_QOH, P_PRICE
FROM PRODUCT
ORDER BY P_PRICE;
```

The output is shown in Figure 7.18. Note that ORDER BY yields an ascending price listing. Comparing the listing in Figure 7.18 to the actual table contents shown earlier in Figure 7.2, you will see that the lowest-priced product is listed first in Figure 7.18, followed by the next lowest-priced product, and so on. However, although ORDER BY produces a sorted output, the actual table contents are unaffected by the ORDER BY operation.

FIGURE 7.18 PRODUCTS SORTED BY PRICE IN ASCENDING ORDER

P_CODE	P_DESCRIPT	P_QOH	P_PRICE
54778-2T	Rat-tail file, 1/8-in. fine	43	4.99
PVC23DRT	PVC pipe, 3.5-in., 8-ft	188	5.87
SM-18277	1.25-in. metal screw, 25	172	6.99
SW-23116	2.5-in. wd. screw, 50	237	8.45
23109-HB	Claw hammer	23	9.95
23114-AA	Sledge hammer, 12 lb.	8	14.40
13-Q2/P2	7.25-in. pwr. saw blade	32	14.99
14-Q1/L3	9.00-in. pwr. saw blade	18	17.49
2238/QPD	B&D cordless drill, 1/2-in.	12	38.95
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15	39.95
1558-QW1	Hrd. cloth, 1/2-in., 3x50	23	43.99
2232/QWE	B&D jigsaw, 8-in. blade	6	99.87
2232/QTY	B&D jigsaw, 12-in. blade	8	109.92
11QER/31	Power painter, 15 psi., 3-nozzle	8	109.99
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	18	119.95
89-WRE-Q	Hicut chain saw, 16 in.	11	256.99

You can add DESC after the attribute to indicate descending order. To produce the listing with products sorted in descending order by price, you would enter:

```
SELECT P_CODE, P_DESCRIPT, P_QOH, P_PRICE
```

```
FROM PRODUCT
ORDER BY P_PRICE DESC;
```

Ordered listings are used frequently. For example, suppose that you want to create a phone directory. It would be helpful if you could produce an ordered sequence (last name, first name, initial) in three stages:

1. ORDER BY last name.
2. Within matching last names, ORDER BY first name.
3. Within matching first and last names, ORDER BY middle initial.

Such a multilevel ordered sequence is known as a cascading order sequence, and it can be created easily by listing several attributes, separated by commas, after the ORDER BY clause.

The cascading order sequence is the basis for any telephone directory. To illustrate a cascading order sequence, use the following SQL command on the EMPLOYEE table:

```
SELECT EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_AREACODE,
EMP_PHONE
FROM EMPLOYEE
ORDER BY EMP_LNAME, EMP_FNAME, EMP_INITIAL;
```

This command yields the results shown in Figure 7.19.

FIGURE 7.19 TELEPHONE LIST QUERY RESULTS

EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_AREACODE	EMP_PHONE
Brandon	Marie	G	901	882-0845
Diante	Jorge	D	615	890-4567
Genkazi	Leighla	W	901	569-0093
Johnson	Edward	E	615	898-4387
Jones	Anne	M	615	898-3456
Kolmycz	George	D	615	324-5456
Lange	John	P	901	504-4430
Lewis	Rhonda	G	615	324-4472
Saranda	Hermine	R	615	324-5505
Smith	George	A	615	890-2984
Smith	George	K	901	504-3339
Smith	Jeanine	K	615	324-7883
Smythe	Melanie	P	615	324-9006
Vandam	Rhett		901	675-8993
Washington	Rupert	E	615	890-4925
Wiesenbach	Paul	R	615	897-4358
Williams	Robert	D	615	890-3220

You can add DESC after the attribute to indicate descending order. To produce the listing with products sorted in descending order by price, you would enter:

```
SELECT P_CODE, P_DESCRIPT, P_QOH, P_PRICE
FROM PRODUCT
ORDER BY P_PRICE DESC;
```

Ordered listings are used frequently. For example, suppose that you want to create a phone directory. It would be helpful if you could produce an ordered sequence (last name, first name, initial) in three stages:

1. ORDER BY last name.
2. Within matching last names, ORDER BY first name.
3. Within matching first and last names, ORDER BY middle initial.

Such a multilevel ordered sequence is known as a cascading order sequence, and it can be created easily by listing several attributes, separated by commas, after the ORDER BY clause.

The cascading order sequence is the basis for any telephone directory. To illustrate a cascading order sequence, use the following SQL command on the EMPLOYEE table:

```
SELECT EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_AREACODE,
EMP_PHONE
FROM EMPLOYEE
ORDER BY EMP_LNAME, EMP_FNAME, EMP_INITIAL;
```

This command yields the results shown in Figure 7.19.

FIGURE 7.20 ORDERING BY A DERIVED ATTRIBUTE

P_CODE	P_DESCRIPT	V_CODE	TOTAL
PVC23DRT	PVC pipe, 3.5-in., 8-ft		1103.56
23114-AA	Sledge hammer, 12 lb.		115.20
SM-18277	1.25-in. metal screw, 25	21225	1202.28
23109-HB	Claw hammer	21225	228.85
SW-23116	2.5-in. wd. screw, 50	21231	2002.65
13-Q2/P2	7.25-in. pwr. saw blade	21344	479.68
14-Q1/L3	9.00-in. pwr. saw blade	21344	314.82
54778-2T	Rat-tail file, 1/8-in. fine	21344	214.57
1558-QVVI	Hrd. cloth, 1/2-in., 3x50	23119	1011.77
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	23119	599.25
89-WRE-Q	Hicut chain saw, 16 in.	24288	2826.89
2232/QTY	B&D jigsaw, 12-in. blade	24288	879.36
2232/QWE	B&D jigsaw, 8-in. blade	24288	599.22
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	25595	2159.10
11QER/31	Power painter, 15 psi., 3-nozzle	25595	879.92
2238/QPD	B&D cordless drill, 1/2-in.	25595	467.40

Lab 3.5. WHERE Clause Options

In this section, you learn how to fine-tune the SELECT command by adding restrictions to the search criteria. When coupled with appropriate search conditions, SELECT is an incredibly powerful tool that enables you to transform data into information. For example, you can create queries that can answer questions such as these: “What products were supplied by a particular vendor?,” “Which products are

priced below $10?," and "How many products supplied by a given vendor were sold between January 5, 2018, and March 20, 2018?"

Selecting Rows with Conditional Restrictions

You can select partial table contents by placing restrictions on the rows to be included in the output. Use the WHERE clause to add conditional restrictions to the SELECT statement that limit the rows returned by the query. The following syntax enables you to specify which rows to select:

```
SELECT columnlist
FROM tablelist
[WHERE conditionlist ]
[ORDER BY columnlist [ASC | DESC] ];
```

The SELECT statement retrieves all rows that match the specified condition(s)— also known as the conditional criteria—you specified in the WHERE clause. The conditionlist in the WHERE clause of the SELECT statement is represented by one or more conditional expressions, separated by logical operators. The WHERE clause is optional. If no rows match the specified criteria in the WHERE clause, you see a blank screen or a message that tells you no rows were retrieved. For example, consider the following query:

```
SELECT P_DESCRIPT, P_QOH, P_PRICE, V_CODE
FROM PRODUCT
WHERE V_CODE = 21344;
```

This query returns the description, quantity on hand, price, and vendor code for products with a vendor code of 21344, as shown in Figure 7.21.

FIGURE 7.21 SELECTED PRODUCT ATTRIBUTES FOR VENDOR CODE 21344

P_DESCRIPT	P_QOH	P_PRICE	V_CODE
7.25-in. pwr. saw blade	32	14.99	21344
9.00-in. pwr. saw blade	18	17.49	21344
Rat-tail file, 1/8-in. fine	43	4.99	21344

Numerous conditional restrictions can be placed on the selected table contents. For example, the comparison operators shown in Table 7.6 can be used to restrict output. Note that there are two options for not equal to. Both <> and != are well supported and perform the same function.

TABLE 7.6

COMPARISON OPERATORS

SYMBOL	MEANING
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> or !=	Not equal to

The following example uses one of the not equal to operators:

```
SELECT P_DESCRIPT, P_QOH, P_PRICE, V_CODE
FROM PRODUCT
WHERE V_CODE <> 21344;
```

The output, shown in Figure 7.23, lists all of the rows for which the vendor code is not 21344.

FIGURE 7.23 PRODUCT ATTRIBUTES FOR VENDOR CODES OTHER THAN 21344

P_DESCRIPT	P_QOH	P_PRICE	V_CODE
Power painter, 15 psi., 3-nozzle	8	109.99	25595
Hrd. cloth, 1/4-in., 2x50	15	39.95	23119
Hrd. cloth, 1/2-in., 3x50	23	43.99	23119
B&D jigsaw, 12-in. blade	8	109.92	24288
B&D jigsaw, 8-in. blade	6	99.87	24288
B&D cordless drill, 1/2-in.	12	38.95	25595
Claw hammer	23	9.95	21225
Hicut chain saw, 16 in.	11	256.99	24288
1.25-in. metal screw, 25	172	6.99	21225
2.5-in. wd. screw, 50	237	8.45	21231
Steel matting, 4'x8'x1/6", .5" mesh	18	119.95	25595

Note that, in Figure 7.23, rows with nulls in the V_CODE column (see Figure 7.2) are not included in the SELECT command's output. The following command sequence:

```
SELECT P_DESCRIPT, P_QOH, P_MIN, P_PRICE
FROM PRODUCT
WHERE P_PRICE <= 10;
```

yields the output shown in Figure 7.24.

FIGURE 7.24 SELECT PRODUCT TABLE ATTRIBUTES WITH A P_PRICE RESTRICTION

P_DESCRIPT	P_QOH	P_MIN	P_PRICE
Claw hammer	23	10	9.95
Rat-tail file, 1/8-in. fine	43	20	4.99
PVC pipe, 3.5-in., 8-ft	188	75	5.87
1.25-in. metal screw, 25	172	75	6.99
2.5-in. wd. screw, 50	237	100	8.45

Using Comparison Operators on Dates

Date procedures are often more software-specific than other SQL procedures. For example, the query to list all of the rows in which the inventory stock dates occur on or after January 20, 2018, looks like this:

```
SELECT P_DESCRIPT, P_QOH, P_MIN, P_PRICE, P_INDATE
```

```
FROM PRODUCT
WHERE P_INDATE >= '20-Jan-2018';
```

FIGURE 7.26 SELECTED PRODUCT TABLE ATTRIBUTES: DATE RESTRICTION

P_DESCRIPT	P_QOH	P_MIN	P_PRICE	P_INDATE
B&D cordless drill, 1/2-in.	12	5	38.95	20-Jan-18
Claw hammer	23	10	9.95	20-Jan-18
Hicut chain saw, 16 in.	11	5	256.99	07-Feb-18
PVC pipe, 3.5-in., 8-ft	188	75	5.87	20-Feb-18
1.25-in. metal screw, 25	172	75	6.99	01-Mar-18
2.5-in. wd. screw, 50	237	100	8.45	24-Feb-18

Logical Operators: AND, OR, and NOT

In the real world, a search of data normally involves multiple conditions. For example, when you are buying a new house, you look for a certain area, a certain number of bedrooms, bathrooms, stories, and so on. In the same way, SQL allows you to include multiple conditions in a query through the use of logical operators. The logical operators are AND, OR, and NOT. For example, if you want a list of the table contents for either the V_CODE = 21344 or the V_CODE = 24288, you can use the OR logical operator, as in the following command sequence:

```
SELECT P_DESCRIPT, P_QOH, P_PRICE, V_CODE
FROM PRODUCT
WHERE V_CODE = 21344 OR V_CODE = 24288;
```

This command generates the six rows shown in Figure 7.27 that match the logical restriction.

FIGURE 7.27 THE LOGICAL OR

P_DESCRIPT	P_QOH	P_PRICE	V_CODE
7.25-in. pwr. saw blade	32	14.99	21344
9.00-in. pwr. saw blade	18	17.49	21344
B&D jigsaw, 12-in. blade	8	109.92	24288
B&D jigsaw, 8-in. blade	6	99.87	24288
Rat-tail file, 1/8-in. fine	43	4.99	21344
Hicut chain saw, 16 in.	11	256.99	24288

The logical operator AND has the same SQL syntax requirement as OR. The following command generates a list of all rows for which P_PRICE is greater than $100 and for which P_QOH is less than 20:

```
SELECT P_DESCRIPT, P_QOH, P_PRICE, V_CODE
FROM PRODUCT
WHERE P_PRICE > 100
AND P_QOH < 20;
```

This command produces the output shown in Figure 7.28.

FIGURE 7.28 THE LOGICAL AND

P_DESCRIPT	P_QOH	P_PRICE	V_CODE
Power painter, 15 psi., 3-nozzle	8	109.99	25595
B&D jigsaw, 12-in. blade	8	109.92	24288
Hicut chain saw, 16 in.	11	256.99	24288
Steel matting, 4'x8'x1/6", .5" mesh	18	119.95	25595

You can combine the logical OR with the logical AND to place further restrictions on the output. For example, suppose that you want a table listing for the following conditions:

- The V_CODE is either 25595 or 24288.
- And the P_PRICE is greater than $100.

The following code produces incorrect results. As shown in Figure 7.29, all rows from vendor 25595 are included in the result even though some of the P_PRICE values are less than the required $100. This is because the DBMS executes the AND operator before the OR operator.

```
SELECT P_DESCRIPT, P_PRICE, V_CODE
FROM PRODUCT
WHERE V_CODE = 25595 OR V_CODE = 24288 AND P_PRICE > 100;
```

FIGURE 7.29 INCORRECT COMBINATION OF AND AND OR

P_DESCRIPT	P_PRICE	V_CODE
Power painter, 15 psi., 3-nozzle	109.99	25595
B&D jigsaw, 12-in. blade	109.92	24288
B&D cordless drill, 1/2-in.	38.95	25595
Hicut chain saw, 16 in.	256.99	24288
Steel matting, 4'x8'x1/6", .5" mesh	119.95	25595

The conditions in the WHERE clause can be grouped using parentheses to produce the desired result. The required listing can be produced by using the following:

```
SELECT P_DESCRIPT, P_PRICE, V_CODE
FROM PRODUCT
WHERE (V_CODE = 25595 OR V_CODE = 24288) AND P_PRICE > 100;
```

Note the use of parentheses to combine logical restrictions. Where you place the parentheses depends on how you want the logical restrictions to be executed. Conditions listed within parentheses are always executed first. The preceding query yields the output shown in Figure 7.30.

FIGURE 7.30 CORRECT COMBINATION AND AND OR CONDITIONS

P_DESCRIPT	P_PRICE	V_CODE
Power painter, 15 psi., 3-nozzle	109.99	25595
B&D jigsaw, 12-in. blade	109.92	24288
Hicut chain saw, 16 in.	256.99	24288
Steel matting, 4'x8'x1/6", .5" mesh	119.95	25595

The use of the logical operators OR and AND can become quite complex when numerous restrictions are placed on the query. In fact, a specialty field in mathematics known as Boolean algebra is dedicated to the use of logical operators. The logical operator NOT is used to negate the result of a conditional expression.

That is, in SQL, all conditional expressions evaluate to true or false. If an expression is true, the row is selected; if an expression is false, the row is not selected. The NOT logical operator is typically used to find the rows that do not match a certain condition. For example, if you want to see a listing of all rows for which the vendor code is not 21344, use the following command sequence:

```
SELECT *
FROM PRODUCT
WHERE NOT (V_CODE = 21344);
```

Note that the condition is enclosed in parentheses; that practice is optional, but it is highly recommended for clarity. The logical operator NOT can be combined with AND and OR.