Chapter 3

How to retrieve data from a single table

Objectives

Applied

- Code SELECT statements that require any of the language elements presented in this chapter.

Knowledge

- Distinguish between the base table values and the calculated values in SELECT statements.
- Describe the use of a column alias.
- Describe the use of the concatenation operator in string expressions.
- Describe the order of precedence and the use of parentheses for arithmetic expressions.
- Describe the use of scalar functions.
- Describe the use of the Dual table, the DISTINCT keyword, and the ROWNUM pseudo column.

Objectives (continued)

Knowledge

- Describe the use of comparison operators, logical operators, and parentheses in WHERE clauses.
- Describe the use of the IN, BETWEEN, and LIKE operators in WHERE clauses.
- Describe the use of IS NULL in a WHERE clause.
- Describe the use of column names, column aliases, calculated values, and column numbers in ORDER BY clauses.

The simplified syntax of the SELECT statement

```
SELECT select_list
FROM table_source
[WHERE search_condition]
[ORDER BY order_by_list]
```

The four clauses of the SELECT statement

- SELECT
- FROM
- WHERE
- ORDER BY

A simple SELECT statement

```
SELECT *
FROM invoices
```

	INVOICE_ID	VENDOR_ID	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL	PAYMENT_TOTAL	CREDIT_TOTAL
1	1	34	QP58872	25-FEB-08	116.54	116.54	0
2	2	34	Q545443	14-MAR-08	1083.58	1083.58	0
3	3	110	P-0608	11-APR-08	20551.18	0	1200
4	4	110	P-0259	16-APR-08	26881.4	26881.4	0

```
(114 rows selected)
```

A SELECT statement that retrieves and sorts

```
SELECT invoice_number, invoice_date, invoice_total
FROM invoices
ORDER BY invoice_total
```

	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL
1	25022117	24-MAY-08	6
2	24863706	27-MAY-08	6
3	24780512	29-MAY-08	6
4	21-4748363	09-MAY-08	9.95

(114 rows selected)

A SELECT statement that retrieves a calculated value

```
SELECT invoice_id, invoice_total,
       (credit_total + payment_total) AS total_credits
FROM invoices
WHERE invoice_id = 17
```

	INVOICE_ID	INVOICE_TOTAL	TOTAL_CREDITS
1	17	356.48	356.48

A SELECT statement that retrieves all invoices between given dates

```
SELECT invoice_number, invoice_date, invoice_total
FROM invoices
WHERE invoice_date BETWEEN '01-MAY-2008'
                       AND '31-MAY-2008'
ORDER BY invoice_date
```

	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL
1	7548906-20	01-MAY-08	27
2	4-321-2596	01-MAY-08	10
3	4-327-7357	01-MAY-08	162.75
4	4-342-8069	01-MAY-08	10

```
(70 rows selected)
```

A SELECT statement that returns an empty result set

```
SELECT invoice_number, invoice_date, invoice_total
FROM invoices
WHERE invoice_total > 50000
```

The expanded syntax of the SELECT clause

```
SELECT [ALL|DISTINCT]
    column_specification [[AS] result_column]
    [, column_specification [[AS] result_column]] ...
```

Five ways to code column specifications

- All columns in base table
- Column name in base table
- Concatenation
- Calculation
- Scalar function

Column specifications that use base table values

The * is used to retrieve all columns

```
SELECT *
```

Column names are used to retrieve specific columns

```
SELECT vendor_name, vendor_city, vendor_state
```

Column specifications that use calculated values

An arithmetic expression that calculates balance_due

```
SELECT invoice_number,
    invoice_total – payment_total – credit_total
    AS balance_due
```

A string expression that derives full_name

```
SELECT first_name || ' ' || last_name AS full_name
```

Two SELECT statements that name the columns

A SELECT statement that uses the AS keyword

```
-- DATE is a reserved keyword.
-- As a result, it must be enclosed in quotations.
SELECT invoice_number AS "Invoice Number",
       invoice_date AS "Date",
       invoice_total AS total
FROM invoices
```

A SELECT statement that omits the AS keyword

```
SELECT invoice_number "Invoice Number",
       invoice_date "Date",
       invoice_total total
FROM invoices
```

The result set for both SELECT statements

	Invoice Number	Date	TOTAL
1	QP58872	25-FEB-08	116.54
2	Q545443	14-MAR-08	1083.58
3	P-0608	11-APR-08	20551.18
4	P-0259	16-APR-08	26881.4
5	MABO1489	16-APR-08	936.93

A SELECT statement that doesn't provide a name for a calculated column

```
SELECT invoice_number, invoice_date, invoice_total,
    invoice_total – payment_total – credit_total
FROM invoices
```

	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL	INVOICE_TOTAL-PAYMENT_TOTAL-CREDIT_TOTAL
1	QP58872	25-FEB-08	116.54	0
2	Q545443	14-MAR-08	1083.58	0
3	P-0608	11-APR-08	20551.18	19351.18
4	P-0259	16-APR-08	26881.4	0
5	MABO1489	16-APR-08	936.93	0

How to concatenate string data

```
SELECT vendor_city, vendor_state,
       vendor_city || vendor_state
FROM vendors
```

	VENDOR_CITY	VENDOR_STATE	VENDOR_CITY||VENDOR_STATE
1	Auburn Hills	MI	Auburn HillsMI
2	Fresno	CA	FresnoCA
3	Olathe	KS	OlatheKS
4	Fresno	CA	FresnoCA
5	East Brunswick	NJ	East BrunswickNJ

How to format string data using literal values

```
SELECT vendor_name,
    vendor_city || ', '
                || vendor_state
                || ' '
                || vendor_zip_code
    AS address
FROM vendors
```

	VENDOR_NAME	ADDRESS
1	Data Reproductions Corp	Auburn Hills, MI 48326
2	Executive Office Products	Fresno, CA 93710
3	Leslie Company	Olathe, KS 66061
4	Retirement Plan Consultants	Fresno, CA 93704
5	Simon Direct Inc	East Brunswick, NJ 08816

How to include apostrophes in literal values

```
SELECT vendor_name || '''s address: ',
       vendor_city || ', '
                   || vendor_state
                   || ' '
                   || vendor_zip_code
FROM vendors
```

	VENDOR_NAME||'"S ADDRESS:'	VENDOR_CITY||','||VENDOR_STATE||'"||VENDOR_ZIP_CODE
1	Data Reproductions Corp's address:	Auburn Hills, MI 48326
2	Executive Office Products's address:	Fresno, CA 93710
3	Leslie Company's address:	Olathe, KS 66061
4	Retirement Plan Consultants's address:	Fresno, CA 93704
5	Simon Direct Inc's address:	East Brunswick, NJ 08816

Terms to know

- String expression
- Literal value
- String literal (string constant)
- Concatenation operator

The arithmetic operators in order of precedence

*	Multiplication
/	Division
+	Addition
-	Subtraction

A SELECT statement that calculates balance due

```
SELECT invoice_total, payment_total, credit_total,
       invoice_total – payment_total – credit_total
           AS balance_due
FROM invoices
```

	INVOICE_TOTAL	PAYMENT_TO...	CREDIT_TOTAL	BALANCE_DUE
1	116.54	116.54	0	0
2	1083.58	1083.58	0	0
3	20551.18	0	1200	19351.18
4	26881.4	26881.4	0	0
5	936.93	936.93	0	0

A SELECT statement that uses parentheses

```
SELECT invoice_id,
    invoice_id + 7 * 3 AS order_of_precedence,
    (invoice_id + 7) * 3 AS add_first
FROM invoices
ORDER BY invoice_id
```

	INVOICE_ID	ORDER_OF_PRECEDENCE	ADD_FIRST
1	1	22	24
2	2	23	27
3	3	24	30
4	4	25	33
5	5	26	36

What determines the sequence of operations

- Order of precedence
- Parentheses

A SELECT statement that uses SUBSTR

```
SELECT vendor_contact_first_name,
vendor_contact_last_name,
    SUBSTR(vendor_contact_first_name, 1, 1) ||
    SUBSTR(vendor_contact_last_name, 1, 1) AS initials
FROM vendors
```

	VENDOR_CONTACT_FIRST_NAME	VENDOR_CONTACT_LAST_NAME	INITIALS
1	Cesar	Arodondo	CA
2	Rachael	Danielson	RD
3	Zev	Alondra	ZA
4	Salina	Edgardo	SE
5	Daniel	Bradlee	DB

A SELECT statement that uses TO_CHAR

```
SELECT 'Invoice: # '
       || invoice_number
       || ', dated '
       || TO_CHAR(payment_date, 'MM/DD/YYYY')
       || ' for $'
       || TO_CHAR(payment_total)
       AS "Invoice Text"
FROM invoices
```

	Invoice Text
1	Invoice: # QP58872, dated 04/11/2006 for $116.54
2	Invoice: # Q545443, dated 05/14/2006 for $1083.58
3	Invoice: # P-0608, dated for $0
4	Invoice: # P-0259, dated 05/12/2006 for $26881.4
5	Invoice: # MABO1489, dated 05/13/2006 for $936.93

A SELECT statement that uses the SYSDATE and ROUND functions

```
SELECT invoice_date,
    SYSDATE AS today,
    ROUND(SYSDATE - invoice_date)  AS invoice_age_in_days
FROM invoices
```

	INVOICE_DATE	TODAY	INVOICE_AGE_IN_DAYS
1	18-JUL-08	01-AUG-08	15
2	20-JUN-08	01-AUG-08	43
3	14-JUN-08	01-AUG-08	49

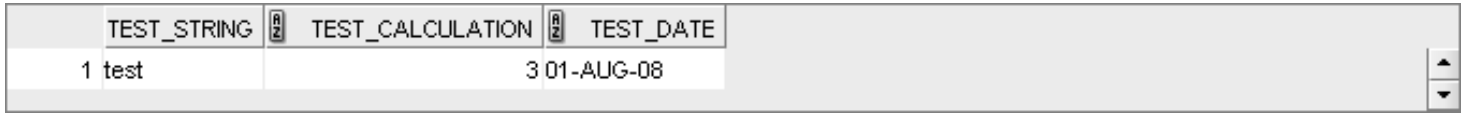
A SELECT statement that uses the MOD function

```
SELECT invoice_id,
    MOD(invoice_id, 10) AS Remainder
FROM invoices
ORDER BY invoice_id
```

	INVOICE_ID	REMAINDER
9	9	9
10	10	0
11	11	1

A SELECT statement that uses the Dual table

```
SELECT 'test'  AS test_string,
       10-7    AS test_calculation,
       SYSDATE AS test_date
FROM Dual
```



	TEST_STRING	TEST_CALCULATION	TEST_DATE
1	test	3	01-AUG-08

A SELECT statement that returns all rows

```
SELECT vendor_city, vendor_state
FROM vendors
ORDER BY vendor_city
```

	VENDOR_CITY	VENDOR_STATE
1	Anaheim	CA
2	Anaheim	CA
3	Ann Arbor	MI
4	Auburn Hills	MI
5	Boston	MA

```
(122 rows selected)
```

A SELECT statement with no duplicate rows

```
SELECT DISTINCT vendor_city, vendor_state
FROM vendors
ORDER BY vendor_city
```

	VENDOR_CITY	VENDOR_STATE
1	Anaheim	CA
2	Ann Arbor	MI
3	Auburn Hills	MI
4	Boston	MA
5	Brea	CA

```
(53 rows selected)
```

A SELECT statement that uses the ROWNUM pseudo column to limit the number of rows

```
SELECT vendor_id, invoice_total
FROM invoices
WHERE ROWNUM <= 5
```

	VENDOR_ID	INVOICE_TOTAL
1	34	116.54
2	34	1083.58
3	110	20551.18
4	110	26881.4
5	81	936.93

A SELECT statement that sorts the result set after the WHERE clause

```
SELECT vendor_id, invoice_total
FROM invoices
WHERE ROWNUM <= 5
ORDER BY invoice_total DESC
```

	VENDOR_ID	INVOICE_TOTAL
1	110	26881.4
2	110	20551.18
3	34	1083.58
4	81	936.93
5	34	116.54

A SELECT statement that sorts the result set before the WHERE clause

```
SELECT vendor_id, invoice_total
FROM (SELECT * FROM invoices
      ORDER BY invoice_total DESC)
WHERE ROWNUM <= 5
```

	VENDOR_ID	INVOICE_TOTAL
1	110	37966.19
2	110	26881.4
3	110	23517.58
4	72	21842
5	110	20551.18

The syntax of the WHERE clause with comparison operators

```
WHERE expression_1 operator expression_2
```

The comparison operators

- =
- >
- <
- <=
- >=
- <>

Examples of WHERE clauses that retrieve…

Vendors located in Iowa

```
WHERE vendor_state = 'IA'
```

Invoices with a balance due (two variations)

```
WHERE invoice_total – payment_total – credit_total > 0
```

```
WHERE invoice_total > payment_total + credit_total
```

Vendors with names from A to L

```
WHERE vendor_name < 'M'
```

Invoices on or before a specified date

```
WHERE invoice_date <= '31-MAY-08'
```

Invoices on or after a specified date

```
WHERE invoice_date >= '01-MAY-08'
```

Invoices with credits that don't equal zero

```
WHERE credit_total <> 0
```

Warning about date comparisons

- All DATE data types include both a date and time.
- The value returned by the SYSDATE function includes a date and a time.
- When you code a date literal like '31-May-2008', the time defaults to 00:00:00 on a 24-hour clock.
- If you ignore the times, a date comparison may not yield the results you expect.

Notes

- Chapter 8 provides solutions for this problem.

The syntax of the WHERE clause with logical operators

```
WHERE [NOT] search_condition_1
    {AND|OR} [NOT] search_condition_2 ...
```

Examples of queries using logical operators

A search condition that uses the AND operator

```
WHERE vendor_state = 'NJ' AND vendor_city = 'Springfield'
```

A search condition that uses the OR operator

```
WHERE vendor_state = 'NJ' OR vendor_city = 'Pittsburgh'
```

A search condition that uses the NOT operator

```
WHERE NOT (invoice_total >= 5000
      OR NOT invoice_date <= '01-JUL-2008')
```

The same condition rephrased to eliminate NOT

```
WHERE invoice_total < 5000
      AND invoice_date <= '01-JUL-2008'
```

A compound condition without parentheses

```
SELECT invoice_number, invoice_date, invoice_total
FROM invoices
WHERE invoice_date > '01-MAY-2008' OR invoice_total > 500
    AND invoice_total – payment_total – credit_total > 0
ORDER BY invoice_number
```

	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL
1	0-2058	08-MAY-08	37966.19
2	0-2060	08-MAY-08	23517.58
3	0-2436	07-MAY-08	10976.06

```
(91 rows selected)
```

The order of precedence for compound conditions

- NOT
- AND
- OR

The same compound condition with parentheses

```
WHERE (invoice_date > '01-MAY-2008'
       OR invoice_total > 500)
    AND invoice_total – payment_total – credit_total > 0
ORDER BY invoice_number
```

	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL
1	0-2436	07-MAY-08	10976.06
2	109596	14-JUN-08	41.8
3	111-92R-10092	04-JUN-08	46.21

(39 rows selected)

The syntax of the WHERE clause with the IN operator

```
WHERE test_expression
  [NOT] IN ({subquery|expression_1 [, expression_2]...})
```

Examples of the IN operator

The IN operator with a list of numeric literals

```
WHERE terms_id IN (1, 3, 4)
```

The IN operator preceded by NOT

```
WHERE vendor_state NOT IN ('CA', 'NV', 'OR')
```

The IN operator with a subquery

```
WHERE vendor_id IN
    (SELECT vendor_id
     FROM invoices
     WHERE invoice_date = '01-MAY-2008')
```

The syntax of the WHERE clause with the BETWEEN operator

```
WHERE test_expression
    [NOT] BETWEEN begin_expression AND end_expression
```

Examples of the BETWEEN operator

The BETWEEN operator with literal values

```
WHERE invoice_date
    BETWEEN '01-MAY-2008' AND '31-MAY-2008'
```

The BETWEEN operator preceded by NOT

```
WHERE vendor_zip_code NOT BETWEEN 93600 AND 93799
```

The BETWEEN operator with a calculated value

```
WHERE invoice_total – payment_total – credit_total
    BETWEEN 200 AND 500
```

The BETWEEN operator with upper and lower limits

```
WHERE invoice_due_date BETWEEN SYSDATE AND (SYSDATE + 30)
```

The syntax of the WHERE clause with the LIKE operator

```
WHERE match_expression [NOT] LIKE pattern
```

Wildcard symbols

%

_

WHERE clauses that use the LIKE operator

Example 1

```
WHERE vendor_city LIKE 'SAN%'
```

Cities that will be retrieved

"San Diego" and "Santa Ana"

Example 2

```
WHERE vendor_name LIKE 'COMPU_ER%'
```

Vendors that will be retrieved

"Compuserve" and "Computerworld"

The syntax of the WHERE clause with the Is null condition

```
WHERE expression IS [NOT] NULL
```

The contents of the Null_Sample table

```
SELECT *
FROM null_sample
```

	INVOICE_ID	INVOICE_TOTAL
1	1	125
2	2	0
3	3	(null)
4	4	2199.99
5	5	0

A SELECT statement that retrieves rows with zero values

```
SELECT *
FROM null_sample
WHERE invoice_total = 0
```

	INVOICE_ID	INVOICE_TOTAL
1	2	0
2	5	0

A SELECT statement that retrieves rows with non-zero values

```
SELECT *
FROM null_sample
WHERE invoice_total <> 0
```

	INVOICE_ID	INVOICE_TOTAL
1	1	125
2	4	2199.99

A SELECT statement that retrieves rows with null values

```
SELECT *
FROM null_sample
WHERE invoice_total IS NULL
```

	INVOICE_ID	INVOICE_TOTAL
1	3	(null)

A SELECT statement that retrieves rows without null values

```
SELECT *
FROM null_sample
WHERE invoice_total IS NOT NULL
```

	INVOICE_ID	INVOICE_TOTAL
1	1	125
2	2	0
3	4	2199.99
4	5	0

The expanded syntax of the ORDER BY clause

```
ORDER BY expression [ASC|DESC]
    [, expression [ASC|DESC]] ...
```

An ORDER BY clause that sorts by one column

```
SELECT vendor_name,
    vendor_city || ', ' || vendor_state || ' ' ||
vendor_zip_code AS address
FROM vendors
ORDER BY vendor_name
```

	VENDOR_NAME	ADDRESS
1	ASC Signs	Fresno, CA 93703
2	AT&T	Phoenix, AZ 85062
3	Abbey Office Furnishings	Fresno, CA 93722

The default sequence for an ascending sort

- Special characters
- Capital letters
- Lowercase letters
- Null values

Notes

- This causes problems when sorting mixed-case columns.
- Chapter 8 provides the solutions.

An ORDER BY clause that sorts by one column in descending sequence

```
SELECT vendor_name,
    vendor_city || ', ' || vendor_state || ' ' ||
vendor_zip_code AS address
FROM vendors
ORDER BY vendor_name DESC
```

	VENDOR_NAME	ADDRESS
1	Zylka Design	Fresno, CA 93711
2	Zip Print & Copy Center	Fresno, CA 93777
3	Zee Medical Service Co	Washington, IA 52353

An ORDER BY clause that sorts by three columns

```
SELECT vendor_name,
    vendor_city || ', ' || vendor_state || ' ' ||
        vendor_zip_code AS address
FROM vendors
ORDER BY vendor_state, vendor_city, vendor_name
```

	VENDOR_NAME	ADDRESS
1	AT&T	Phoenix, AZ 85062
2	Computer Library	Phoenix, AZ 85023
3	Wells Fargo Bank	Phoenix, AZ 85038
4	Aztek Label	Anaheim, CA 92807
5	Blue Shield of California	Anaheim, CA 92850
6	Diversified Printing & Pub	Brea, CA 92621
7	ASC Signs	Fresno, CA 93703

An ORDER BY clause that uses an alias

```
SELECT vendor_name,
    vendor_city || ', ' || vendor_state || ' ' ||
        vendor_zip_code AS address
FROM vendors
ORDER BY address, vendor_name
```

	VENDOR_NAME	ADDRESS
1	Aztek Label	Anaheim, CA 92807
2	Blue Shield of California	Anaheim, CA 92850
3	Malloy Lithographing Inc	Ann Arbor, MI 48106
4	Data Reproductions Corp	Auburn Hills, MI 48326

An ORDER BY clause that uses an expression

```
SELECT vendor_name,
    vendor_city || ', ' || vendor_state || ' ' ||
        vendor_zip_code AS address
FROM vendors
ORDER BY vendor_contact_last_name
    || vendor_contact_first_name
```

	VENDOR_NAME	ADDRESS
1	Dristas Groom & McCormick	Fresno, CA 93720
2	Internal Revenue Service	Fresno, CA 93888
3	US Postal Service	Madison, WI 53707
4	Yale Industrial Trucks-Fresno	Fresno, CA 93706

An ORDER BY clause that uses column positions

```
SELECT vendor_name,
    vendor_city || ', ' || vendor_state || ' ' ||
        vendor_zip_code AS address
FROM vendors
ORDER BY 2, 1
```

	VENDOR_NAME	ADDRESS
1	Aztek Label	Anaheim, CA 92807
2	Blue Shield of California	Anaheim, CA 92850
3	Malloy Lithographing Inc	Ann Arbor, MI 48106
4	Data Reproductions Corp	Auburn Hills, MI 48326