Working with Oracle SQL

Chapter 2:
SQL Query Syntax

Chapter Objectives

In this chapter, we will discuss:

- Building basic `SELECT` statements
- Using the `WHERE` clause and the comparison operators
- Sorting the result set using the `ORDER BY` clause

Chapter Concepts

➤ **Building Basic `SELECT` Statements**

The `WHERE` Clause

The `ORDER BY` Clause

Chapter Summary

The Nature of SQL Statements

- `SELECT`, like all SQL statements, is a non-procedural, descriptive data access command
  - We describe WHAT we want, not HOW to do it
- The description is implemented by key words, followed by clauses that modify the key word
  - The clauses can have one or more entries
  - Not all key words must appear in the statement
    - Only `SELECT` and `FROM` are always required
      - Even if the statement does not need data from a table

Structure of the `SELECT` Statement

```
SELECT
     column or expression, column or expression …
FROM
     table
WHERE
     condition 1  AND/OR condition 2 …
ORDER BY
     column or expression or column alias or position, …
```

The `SELECT` List

- The data to be returned is defined in the `SELECT` list
- Data elements are comma delimited
- The most common data elements are columns from some table
  - Anything that is in scope can be `SELECT`ed
    - Any column in a table in the `FROM` clause
    - Literals
    - Expressions
    - Function calls returning data
- By default, Oracle will use the name of the column for the heading
  - Frontend tools format the width of the data based upon the definition of the column stored in the Data Dictionary

SELECT List Examples

- A column, an expression, or a literal can be SELECTed

```
SELECT last_name, salary, salary * 12, 'Wow', 1/8 FROM employees;

LAST_NAME                     SALARY   SALARY*12 'WO        1/8
------------------------- ---------- ---------- --- ----------
King                           24000      288000 Wow       .125
Kochhar                        17000      204000 Wow       .125
De Haan                        17000      204000 Wow       .125
```

- A literal can be selected from any table, but will be returned in a multiple number of rows

```
SELECT 1/8 FROM employees;

       1/8
----------
      .125
      .125 ...
107 rows selected.
```

The Dummy Table, DUAL

- Oracle provides a table named DUAL to provide a workaround for the ANSI requirement that every SELECT statement *must* have a FROM clause
  - Useful when the information is not in a particular table

```
SELECT 1/8 FROM DUAL;

       1/8
----------
      .125
```

SELECT All Columns from a Table

- First, determine the columns using the DESCRIBE command:

```
desc jobs

 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 JOB_ID                                    NOT NULL VARCHAR2(10)
 JOB_TITLE                                 NOT NULL VARCHAR2(35)
 MIN_SALARY                                         NUMBER(6)
 MAX_SALARY                                         NUMBER(6)
```

- Then, issue the SELECT statement

```
SELECT * FROM jobs;

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
AD_PRES    President                                20000      40000
AD_VP      Administration Vice President            15000      30000
AD_ASST    Administration Assistant                  3000       6000
FI_MGR     Finance Manager                           8200      16000
```

- Notice the presentation sequence of the columns

Column Alias

- A column alias can also be used to override the default heading name
  - Syntax:
    - `column AS column_alias, …`
      - `AS` is an optional key word
    - `column column_alias , …`
  - Prefer using `AS` since it makes your intentions clear and avoids missing commas
  - The column alias can be a string with no spaces, or be bound within double quotes
    - Double quotes will respect the case of the string

```
SELECT job_title AS position, job_title AS "Position Title" FROM jobs;

POSITION                            Position Title
----------------------------------- -----------------------------
President                           President
Administration Vice President       Administration Vice President
```

NULLs

- The data value can be `NULL`
  - Meaning, a value has not been assigned
- If the value of a column is `NULL`:
  - Then it will be displayed as blank in script output
  - Or as (null) in the results window tabular view

```
SELECT city, state_province, country_id FROM locations;

CITY                           STATE_PROVINCE            CO
------------------------------ ------------------------- --
Roma                                                     IT
Venice                                                   IT
Tokyo                          Tokyo Prefecture          JP
Hiroshima                                                JP
Southlake                      Texas                     US
South San Francisco            California                US
```

`ALL` or `DISTINCT`

- Relational theory mandates that all tuples (column values) in a set be unique
  - Not the default with SQL
- The implied set is defined by `ALL`
  - `SELECT ALL` …
- If we only want the unique rows, we add `DISTINCT` (or `UNIQUE`) to the `SELECT` List
  - `SELECT DISTINCT` …
  - Distinct applies to the *entire* `SELECT` list
    - Not just the column it appears in front of

ALL or DISTINCT Example

```
SELECT country_id
FROM locations;

CO
--
IT
IT
JP
JP
US
US
US
US
CA
CA
CN
```

```
IN
AU
SG
UK
UK
UK
DE
BR
CH
CH
NL
MX

23 rows selected.
```

```
SELECT DISTINCT country_id
FROM locations;

CO
--
AU
BR
CA
CH
CN
DE
IN
IT
JP
MX
NL
SG
UK
US

14 rows selected.
```

DISTINCT What?

- Remember that the DISTINCT applies to the entire select list

```
SELECT DISTINCT country_id, city FROM locations;


CO CITY
-- ------------------------------
AU Sydney
BR Sao Paulo
CA Toronto
CA Whitehorse
CH Bern
CH Geneva
CN Beijing
DE Munich
IN Bombay
IT Roma
IT Venice
```

Chapter Concepts

Building Basic `SELECT` Statements

The `WHERE` Clause

The `ORDER BY` Clause

Chapter Summary

The WHERE Clause

- Filters rows of data out of the result set
  - Predicated upon condition(s) testing True, False, or NULL
  - NULL, in a condition, always evaluates to False
- Example:
  - Restrict information about JOBS to those with a minimum salary of exactly 4000

```
SELECT *
FROM jobs
WHERE min_salary = 4000;

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
IT_PROG    Programmer                                4000      10000
MK_REP     Marketing Representative                  4000       9000
HR_REP     Human Resources Representative            4000       9000
```

Constructing the Value to Be Tested: Strings

- String literals, sometimes referred to as character literals, are placed in single quote marks:
  - `'This is a string literal'`
  - Any valid character can be part of a string literal
  - Including the single quote, which is escaped by another single quote
  - `'This is Oracle''s character set'`

```
SELECT *
FROM jobs
WHERE job_title = 'Marketing Manager';

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
MK_MAN     Marketing Manager                         9000      15000
```

Case Sensitivity

- String literals ARE case sensitive
  - `'A'` and `'a'` are not the same

```
SELECT *
FROM jobs
WHERE job_title = 'MARKETING MANAGER';

no rows selected
```

- Handling case sensitivity
  - Some shops have the standard that ALL strings stored in the database must be in UPPERcase
    - May not be feasible
  - Can also be handled in SQL statement with functions (covered later)

Comparison Operators

- The test does not have to always be equal to (=)
- Other comparison operators include:
  - Not equal to specified as <> OR !=
  - Greater than > , less than <
  - Greater than or equal to >= , less than or equal to <=

```
SELECT *
FROM jobs
WHERE min_salary <> 4000;

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
AD_PRES    President                                20000      40000
AD_VP      Administration Vice President            15000      30000
AD_ASST    Administration Assistant                  3000       6000
FI_MGR     Finance Manager                           8200      16000
FI_ACCOUNT Accountant                                4200       9000
...
```

More Comparison Operators: BETWEEN

- BETWEEN is used to describe a range of values inclusive of the end values
  - BETWEEN a AND b
  - The operator includes both end points
    - Both value a and value b will test TRUE

```
SELECT *
FROM jobs
WHERE min_salary BETWEEN 3000 AND 4000;

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
AD_ASST    Administration Assistant                  3000       6000
IT_PROG    Programmer                                4000      10000
MK_REP     Marketing Representative                  4000       9000
HR_REP     Human Resources Representative            4000       9000
```

- NOT BETWEEN is the logical opposite
  - The above four rows would NOT be included in the result set

More Comparison Operators: IN

- IN tests to determine if it is in a list of values
  - IN (a,b,c)

```
SELECT *
FROM jobs
WHERE min_salary IN (3000, 4000);

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
AD_ASST    Administration Assistant                  3000       6000
IT_PROG    Programmer                                4000      10000
MK_REP     Marketing Representative                  4000       9000
HR_REP     Human Resources Representative            4000       9000
```

- NOT IN is the logical opposite
  - NOT IN (x,y,z)

More Comparison Operators: `LIKE`

- `LIKE` tests a string for some sequence of characters
  - Character strings are enclosed in single 'quotes'
  - Remember that string literals are case sensitive
- Two wild card characters can be used to test for unspecified values
  - `%` means any value and zero or more characters
  - `_` means any single character
- `NOT LIKE` is the logical opposite
- For example, list the names of all employees whose last name begins with P

```
SELECT first_name, last_name
FROM employees
WHERE last_name LIKE 'P%';

FIRST_NAME           LAST_NAME
-------------------- ---------------------
Karen                Partners
Valli                Pataballa
Joshua               Patel
Randall              Perkins
Hazel                Philtanker
Luis                 Popp
```

Testing for NULL

- NULL is never equal (or not equal) to anything
  - NULL is never less than or greater than any value
  - NULL is never equal to or not equal to itself!
  - Testing against NULL is always false
- Testing to be = NULL is legal syntax
  - But no rows will ever be selected

```
SELECT * FROM jobs WHERE min_salary = NULL;

no rows selected
```

```
SELECT * FROM jobs WHERE min_salary <> NULL;

no rows selected
```

Testing for NULL (continued)

- Must use the comparison operator `IS NULL`

```
SELECT city, state_province, country_id
FROM locations
WHERE state_province IS NULL;

CITY                           STATE_PROVINCE            CO
------------------------------ ------------------------- --
Roma                                                     IT
Venice                                                   IT
Hiroshima                                                JP
Beijing                                                  CN
Singapore                                                SG
London                                                   UK
```

- `IS NOT NULL` is the logical opposite

More than One Condition

- Multiple conditions (Boolean logic) can be constructed
  - Can specify an unlimited number of conditions
    - As long as the relationship between them is stated
  - Need to specify the logical operator: `AND`, `OR`, `NOT`

```
SELECT *
FROM jobs
WHERE min_salary = 3000 OR min_salary = 4000;

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
AD_ASST    Administration Assistant                  3000       6000
IT_PROG    Programmer                                4000      10000
MK_REP     Marketing Representative                  4000       9000
HR_REP     Human Resources Representative            4000       9000
```

Multiple Conditions: AND

- AND means both must be true

```
SELECT *
FROM jobs
WHERE min_salary = 4000 AND max_salary < 10000;

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
MK_REP     Marketing Representative                  4000       9000
HR_REP     Human Resources Representative            4000       9000
```

- Suppose we wanted all the rows other than these?

NOT-ing the Evaluation

- It is legal to NOT the paired conditional logic
  - Similar to applying NOT to a single condition
- Syntax: group the conditions with parentheses and NOT the group

```
SELECT *
FROM jobs
WHERE NOT (min_salary = 4000 AND max_salary < 10000);

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
AD_PRES    President                                20000      40000
AD_VP      Administration Vice President            15000      30000
AD_ASST    Administration Assistant                  3000       6000
```

- Notice that this describes the opposite set
  - All the rows not included before

Specifying the Sequencing of Conditional Evaluation

- By default, Oracle will follow this precedence:
  - `NOT`s evaluated first, then `AND`s, then `OR`s
  - Use parentheses to override the default evaluation order
- Since few people remember the order, favor parentheses for clarity
  - Also improves readability
  - And makes code more robust when being modified
- Question: Do the following statements describe the same data?

```
SELECT * FROM jobs
WHERE job_title = 'Marketing Manager'
OR min_salary = 4000 AND max_salary < 10000 ;


SELECT * FROM jobs
WHERE (job_title = 'Marketing Manager'
OR min_salary = 4000) AND max_salary < 10000 ;
```

Chapter Concepts

Building Basic `SELECT` Statements

The `WHERE` Clause

The `ORDER BY` Clause

Chapter Summary

ORDER BY

- `ORDER BY` is the next clause of the `SELECT` statement
  - Its objective is to sort the result set
  - Does not change any of the data being returned
  - Just the output sequence
    - Otherwise, the data is in a "heap"
  - The rows in the order of Oracle's most efficient retrieval method

Syntax of `ORDER BY`

- Ordering can be specified by column name, column expression, column alias, select list position
  - The first is the primary sort, the second is sorted within the primary
  - The default sequence is `ASC`ending
    - Usually not specified
  - Sort order can be `DESC`ending
  - `NULL`s sorts high
  - The column being sorted on does not have to be in the `SELECT` list
    - Any column in any table in the query can be referenced

Which Job Title Do I NOT Want?

```
SELECT job_title, max_salary
FROM jobs
ORDER BY max_salary;

JOB_TITLE                           MAX_SALARY
----------------------------------- ----------
Stock Clerk                               5000
Purchasing Clerk                          5500
Shipping Clerk                            5500
Administration Assistant                  6000
Stock Manager                             8500
Accountant                                9000
Public Accountant                         9000
Marketing Representative                  9000
Human Resources Representative            9000
Programmer                               10000
Public Relations Representative          10500
Sales Representative                     12000
Purchasing Manager                       15000
Marketing Manager                        15000
Finance Manager                          16000
Accounting Manager                       16000
Sales Manager                            20000
Administration Vice President            30000
President                                40000

19 rows selected.
```

Which Job Title Do I Want?

```
SELECT job_title, max_salary
FROM jobs
ORDER BY max_salary DESC;

JOB_TITLE                           MAX_SALARY
----------------------------------- ----------
President                                40000
Administration Vice President            30000
Sales Manager                            20000
Finance Manager                          16000
Accounting Manager                       16000
Purchasing Manager                       15000
Marketing Manager                        15000
Sales Representative                     12000
Public Relations Representative          10500
Programmer                               10000
Accountant                                9000
Public Accountant                         9000
Human Resources Representative            9000
Marketing Representative                  9000
Stock Manager                             8500
Administration Assistant                  6000
Purchasing Clerk                          5500
Shipping Clerk                            5500
Stock Clerk                               5000

19 rows selected.
```

Ordering on an Expression

- The expression could be repeated …

```
SELECT job_title, max_salary / 12 AS "Monthly Salary"
FROM jobs
WHERE max_salary / 12 > 1000
ORDER BY max_salary / 12 DESC, job_title;

JOB_TITLE                           Monthly Salary
----------------------------------- --------------
President                               3333.33333
Administration Vice President                 2500
Sales Manager                           1666.66667
Accounting Manager                      1333.33333
Finance Manager                         1333.33333
Marketing Manager                             1250
Purchasing Manager                            1250

7 rows selected.
```

Ordering on a Column Alias

- This is legal because the `ORDER BY` happens after the result set has been determined
  - And the column aliases have been applied to the set

```
SELECT job_title, max_salary / 12 AS "Monthly Salary"
FROM jobs
WHERE max_salary / 12 > 1000
ORDER BY "Monthly Salary" DESC, job_title;

JOB_TITLE                           Monthly Salary
----------------------------------- --------------
President                               3333.33333
Administration Vice President                 2500
Sales Manager                           1666.66667
Accounting Manager                      1333.33333
Finance Manager                         1333.33333
Marketing Manager                             1250
Purchasing Manager                            1250

7 rows selected.
```

Ordering by SELECT List Position

- This is still legal but it is not a good practice
  - Useful for ad hoc statements

```
SELECT job_title, max_salary / 12 AS "Monthly Salary"
FROM jobs
WHERE max_salary / 12 > 1000
ORDER BY 2 DESC, job_title;

JOB_TITLE                           Monthly Salary
----------------------------------- --------------
President                               3333.33333
Administration Vice President                 2500
Sales Manager                           1666.66667
Accounting Manager                      1333.33333
Finance Manager                         1333.33333
Marketing Manager                             1250
Purchasing Manager                            1250

7 rows selected.
```

Positioning NULLs: High by Default

- Given the following set of data:

```
DEPARTMENT_NAME                  MANAGER_ID
------------------------------ ----------
Sales                                   145
Executive                               100
Finance                                 108
Accounting                              205
Treasury
Corporate Tax
Control And Credit
```

By default, NULLs sort high

```
SELECT department_name, manager_id
FROM departments
ORDER BY manager_id;

DEPARTMENT_NAME                  MANAGER_ID
------------------------------ ----------
Executive                               100
Finance                                 108
Sales                                   145
Accounting                              205
Treasury
Corporate Tax
Control And Credit
```

```
SELECT department_name, manager_id
FROM departments
ORDER BY manager_id DESC;

DEPARTMENT_NAME                  MANAGER_ID
------------------------------ ----------
Treasury
Corporate Tax
Control And Credit
Accounting                              205
Sales                                   145
Finance                                 108
Executive                               100
```

Forcing the Placement of NULLs

- The options on the ORDER BY clause include:
  - NULLS FIRST and NULLS LAST
  - This forces the NULLS to (positionally) be on the top or bottom
    - Without regard to whether the sort order is ASC or DESC

```
SELECT department_name, manager_id
FROM departments
ORDER BY manager_id DESC  NULLS FIRST;

DEPARTMENT_NAME                MANAGER_ID
------------------------------ ----------
Treasury
Corporate Tax
Control And Credit
Accounting                            205
Sales                                 145
Finance                               108
Executive                             100
```

```
SELECT department_name, manager_id
FROM departments
ORDER BY manager_id DESC  NULLS LAST;

DEPARTMENT_NAME                MANAGER_ID
------------------------------ ----------
Accounting                            205
Sales                                 145
Finance                               108
Executive                             100
Treasury
Corporate Tax
Control And Credit
```

Exercise 2.1: Selecting Data

HANDS-ON EXERCISE

60 min

- Please complete this exercise in your Exercise Manual

Chapter Concepts

Building Basic `SELECT` Statements

The `WHERE` Clause

The `ORDER BY` Clause

➤ **Chapter Summary**

Chapter Summary

In this chapter, we have discussed:

- Building basic `SELECT` statements
- Using the `WHERE` clause and the comparison operators
- Sorting the result set using the `ORDER BY` clause