Working with Oracle SQL

Chapter 8:

SQL Functions

Chapter Objectives

In this chapter, we will:

- Define common datatypes
- Use simple SQL functions
  - Definition
  - Classes of functions
  - Common Single Row (Scalar) Functions
- Using `DATE`-related functions
- Miscellaneous functions

Chapter Concepts

`NUMBER` Datatype

- Used to store fixed or floating-point numbers
- Syntax:

```
NUMBER [(precision, scale)]
```

- Precision
  - Total number of significant digits
  - Optional, defaults to the maximum (38 digits)
- Scale
  - Number of digits after the decimal point
  - Can range from –84 to 127
  - Optional, defaults to zero

- Examples:
  - `NUMBER`
    - 38 total digits (before or after the decimal point)
  - `NUMBER(2)`
    - Two digits before the decimal point and zero digits after
  - `NUMBER(3, 2)`
    - One digit before the decimal point and two digits after
  - `NUMBER(*, 2)`
    - 38 digits of total precision with two digits after the decimal point

`CHAR` Datatypes

- `CHAR` is used to store fixed-length character data
  - Syntax:
    `CHAR [(length)]`
- Length
  - Maximum 2,000 bytes
  - Optional, defaults to 1
  - Values are padded with blanks to the maximum length
- Example:
  - `CHAR` stores one character
  - `CHAR(10)` stores 10 characters for a value of any length
  - Storing `'MIKE'` in this datatype would result in `MIKE` and six blanks

`VARCHAR2` Datatype

- Used to store variable-length character data
  - Syntax:
    `VARCHAR2(length)`
- Length
  - Maximum 4,000 bytes
    - Since Oracle 12c, there is a database option to allow strings as long as 32767 bytes
  - Mandatory
  - Values are not padded; exactly the length of the string is stored
- Example:
  - `VARCHAR2(10)` stores up to 10 characters based on the actual string
  - Storing `'MIKE'` in this datatype would not store extra characters

`DATE` Datatype

- Used to store date and time to the precision of seconds
  - Syntax:

    ```
    DATE
    ```
- Stored internally as an ordered set of seven bytes, representing century, year, month, day, hour, minute, second
  - All `DATE`s contain a date and a time (differs from the standard)
  - If the time is not set, it defaults to midnight
  - If the date is not set, it defaults to the first day of the current month
- Can add and subtract dates
  - `start_date + 1` is one day after start date
  - `end_date - start_date` is the number of days in this period

Date Literals

- Oracle will interpret certain character literals as dates when needed
  - Relies on the default format mask set by `NLS_DATE_FORMAT`
    - Defaults to `'DD-MON-YY'`, e.g., `'21-JAN-01'`
    - This course is set to `'DD-MON-YYYY'`, e.g., `'21-JAN-2001'`
- ***Do not rely on this***
  - You cannot always control the format
  - It is fine for testing, but not production code
- Either use the ANSI standard date format
  - `DATE 'YYYY-MM-DD'`, e.g., `DATE '2001-01-01'`
    - Only useful for setting dates, does not support times
- Or use `TO_DATE(char_literal, format)`
    - E.g., `TO_DATE('98-DEC-25 17:30', 'YY-MON-DD HH24:MI')`

https://docs.oracle.com/cd/B19306_01/server.102/b14200/sql_elements003.htm#BABGIGCJ

Date and Time Format Models

- Commonly used format models for `DATE` and `TIMESTAMP` datatypes

Format Model	Meaning
YYYY	Four-digit year
YY	Two-digit year
MON	Three-character name of month
MM	Two-digit month
DD	Two-digit day
HH	Two-digit hour of day (1–12)
AM	Two character meridian indicator
HH24	Two-digit hour of day (0–23)
MI	Two-digit minute (0–59)
SS	Two-digit seconds (0–59)
FF	Fractional seconds (1–9 digits)

`TIMESTAMP` Datatype

- Extension of the `DATE` datatype that supports fractional seconds
- Syntax:

```
TIMESTAMP [ (precision)]
```

- Precision specifies the number of digits in the fractional part of seconds that will be stored and displayed
  - Can be a number in the range 0 to 9 (default is 6 digits)
- Retrieved and updated based on date and time format models
  - Default format mask is `'DD-MON-YY HH.MI.SS.FF AM'`
  - Changed by setting `NLS_TIMESTAMP_FORMAT` parameter
    - This course is set to `'DD-MON-YYYY HH24:MI:SS.FF'`
    - Example of four-digit precision: `'21-JAN-2001 20:12:10.0250'`
  - ***Do not rely on this***, use ANSI: `TIMESTAMP '2001-01-21 20:12:10.0250'`

Implicit Datatype Conversion

From\to	CHAR	VARCHAR2	DATE	TIMESTAMP	NUMBER
CHAR		Yes	Yes	Yes	Yes
VARCHAR2	Yes		Yes	Yes	Yes
DATE	Yes	Yes		No	No
TIMESTAMP	Yes	Yes	No		No
NUMBER	Yes	Yes	No	No	

- Implicit conversion to dates and timestamp requires that the string be in the default date or timestamp format

Chapter Concepts

Functions

- Manipulate data items and return a result
  - Modify a value
  - Combine values
  - Change value formats
  - Create new values
- Syntax:

```
FUNCTION_NAME(parameter1, parameter2, … parameterN)
```

  - Some functions require no parameters
- Most SQL functions are ANSI compliant
  - There is a standard specification that vendors adhere to
  - They will work the same with any RDBMS that is ANSI compliant
- Oracle, like other vendors, supplies functions that are not ANSI
  - Considered extensions to standard SQL

Classes of Functions

- Functions are classified according to nature of the data they are working on
- Single Row (or scalar) Functions
  - Single-row functions return a single result row for every row of a queried table or view
    - These are the type we will be discussing in this chapter
- Aggregate functions
  - Return a single result row based on groups of rows, rather than on single rows
    - Already covered

Single Row (Scalar) Functions: Types

- There are over 150 Single Row Functions
- We will look at the most commonly used functions
  - Numeric
  - Character or string
  - Date/Time
  - Analytical
  - Miscellaneous
- Others handle explicit conversion of datatypes

Chapter Concepts

Basic Server Datatypes

Introduction to Functions

Scalar Functions

`DATE` Functions

Miscellaneous Functions

Chapter Summary

Numeric Functions

- Accept numeric input and return numeric values
  - Most numeric functions that return `NUMBER` values that are accurate to 38 decimal digits
- The numeric functions are:

```
ABS
ACOS
ASIN
ATAN
ATAN2
BITAND
CEIL
COS
COSH
```

```
EXP
FLOOR
LN
LOG
MOD
NANVL
POWER
REMAINDER
ROUND (number)
```

```
SIGN
SIN
SINH
SQRT
TAN
TANH
TRUNC (number)
WIDTH_BUCKET
```

ROUND: Numeric

- `ROUND(the_numeric_value, the_degree_of_rounding)`
  - If the second parameter is not supplied, then 0 decimal positions is assumed
  - If the second parameter is negative, rounds to the left of the decimal

```
SELECT 123.45
,ROUND(123.45)     AS A1 ,ROUND(123.45,1)  AS B1 ,ROUND(123.45,2)  AS C1
,ROUND(123.45,-1) AS D1 ,ROUND(123.45,-2) AS E1  FROM DUAL;

    123.45         A1         B1         C1         D1         E1
---------- ---------- ---------- ---------- ---------- ----------
    123.45        123      123.5     123.45        120        100
```

```
SELECT 123.55
,ROUND(123.55)     AS A2 ,ROUND(123.55,1)  AS B2 ,ROUND(123.55,2)  AS C2
,ROUND(123.55,-1) AS D2 ,ROUND(123.55,-2) AS E2  FROM DUAL;

    123.55         A2         B2         C2         D2         E2
---------- ---------- ---------- ---------- ---------- ----------
    123.55        124      123.6     123.55        120        100
```

TRUNC: Numeric

- TRUNC(the_numeric_value, the_degree_of_rounding)
  - If the second parameter is not supplied, then 0 decimal positions is assumed
  - If the second parameter is negative, truncates to the left of the decimal

```
SELECT 123.55
      ,TRUNC(123.55)     AS A
      ,TRUNC(123.55,1)   AS B
      ,TRUNC(123.55,2)   AS C
      ,TRUNC(123.55,-1)  AS D
      ,TRUNC(123.55,-2)  AS E
FROM DUAL;

    123.55           A          B           C           D           E
---------- ----------- ---------- ----------- ----------- ----------
    123.55      123.00     123.50      123.55         120         100
```

Character Functions: Returning Characters

- Character functions that return character values
  - Also referred to as string functions
  - The length of the value returned by the function is limited by the maximum length of the datatype returned
- The character functions that return character values are:

```
CHR
CONCAT or ||
INITCAP
LOWER
LPAD
LTRIM
NLS_INITCAP
NLS_LOWER
```

```
NLSSORT
NLS_UPPER
REGEXP_LIKE
REGEXP_REPLACE
REGEXP_SUBSTR
REPLACE
RPAD
RTRIM
```

```
SOUNDEX
SUBSTR
TRANSLATE
TREAT
TRIM
UPPER
```

Character Functions: Concatenating

- `CONCAT(parameter1, parameter2)` returns a single string
- The more useful construction is `||`
  - Can string more than one parameter together
  - Oracle provides implicit datatype conversions
- Example:

```
SELECT first_name || ' ' || last_name || ' was hired on ' || hire_date
FROM employees
WHERE employee_id IN (163,164);

FIRST_NAME||''||LAST_NAME||'WASHIREDON'||HIRE_DATE
-------------------------------------------------------------
Danielle Greene was hired on 19-MAR-99
Mattea Marvins was hired on 24-JAN-00
```

Character Functions: `UPPER` and `LOWER`

- Return the string in the specified case: `UPPER(value)`, `LOWER(value)`

```
SELECT LOWER(first_name) || ' ' || UPPER(last_name) || ' was hired on ' || hire_date
FROM employees
WHERE employee_id IN (163,164);

LOWER(FIRST_NAME)||''||UPPER(LAST_NAME)||'WASHIREDON'||HIRE_DATE
-------------------------------------------------------------------
danielle GREENE was hired on 19-MAR-99
mattea MARVINS was hired on 24-JAN-00
```

- Useful when the case of the character columns is not known, or not consistent

```
SELECT  hire_date FROM employees WHERE last_name = 'GREENE';

no rows selected

SELECT  hire_date FROM employees WHERE UPPER(last_name) = 'GREENE';

HIRE_DATE
-----------
19-MAR-1999
```

Character Functions: SUBSTR

- The SUBSTR functions return a portion of string
- Syntax: SUBSTR(some_string, position, substring_length)
  - Returns the string beginning at position and length of substring_length
    - substring_length defaults to the end of the string
    - If position is negative, then it is relative to the end of the string

```
SELECT country_name
     , SUBSTR(country_name,1,2)    AS A
     , SUBSTR(country_name,1)      AS B
     , SUBSTR(country_name,5,3)    AS C
     , SUBSTR(country_name,-10,3)  AS D
     , SUBSTR(country_name,-4)     AS E
FROM countries
WHERE country_id = 'CH';

COUNTRY_NAM A           B           C           D           E
----------- ----------- ----------- ----------- ----------- ---------
Switzerland Sw          Switzerland zer         wit         land
```

Character Functions: `TRIM`, `LTRIM`, `RTRIM`

- `TRIM` enables you to trim leading or trailing characters (or both) from a character string
- Syntax:
  `TRIM([[LEADING | TRAILING | BOTH ] trim_character FROM] source)`
- Removes consecutive characters matching `trim_character` from specified position
  - `BOTH` is the default
  - If you do not specify `trim_character`, then the default value is a blank space
  - So, `TRIM(column_name)` will remove leading and trailing blank spaces
  - If either parameter is `NULL`, then the function returns `NULL`
  - `TRIM` is an ANSI standard function
- Older Oracle functions are `RTRIM` and `LTRIM`
  - `LTRIM(source, trim_characters)`
  - Can only trim from one side, but can trim more than one character
  - To trim `BOTH`: `LTRIM(RTRIM(source, trim_characters), trim_characters)`

Character Functions: TRIM, LTRIM, RTRIM Examples

```
SELECT job_title
     , TRIM(BOTH     'M'  FROM job_title) AS A
     , TRIM(LEADING  'M'  FROM job_title) AS B
     , TRIM(TRAILING 'R'  FROM job_title) AS C
     , TRIM(TRAILING 'r'  FROM job_title) AS D
FROM jobs
WHERE job_id = 'MK_MAN';

JOB_TITLE           A                   B                   C                   D
------------------- ------------------- ------------------- ------------------- -------------------
Marketing Manager   arketing Manager    arketing Manager    Marketing Manager   Marketing Manage
```

```
SELECT job_title
     , LTRIM(job_title, 'M')                       AS A
     , LTRIM(RTRIM(job_title, 'M'), 'M')           AS B
     , LTRIM(RTRIM(job_title, 'Mare'), 'Mare')     AS C
FROM jobs
WHERE job_id = 'MK_MAN’;

JOB_TITLE           A                   B                   C
------------------- ------------------- ------------------- -------------------
Marketing Manager   arketing Manager    arketing Manager    keting Manag
```

Character Functions: Returning a Number

- Character functions that return number values can take as their argument any character datatype
- The character functions that return number values are:

```
ASCII
INSTR
LENGTH
REGEXP_INSTR
```

Character Functions: `INSTR`

- The INSTR functions search string for substring
- Syntax:

```
INSTR(string, substring, position, occurrence)
```

  - Returns an integer indicating the position `substring` in `string`
    - Value is the position of the first character of `substring` in this occurrence
  - `position` is the character position in `string` where the search begins
    - If negative, then `INSTR` counts and searches backward from the end of `string`
    - Default is 1
  - `occurrence` indicates which occurrence of `substring` Oracle should search for
    - Must be positive, default is 1

Character Functions: INSTR Example

- Search for the strings of Marketing Manager

```
SELECT job_title
     , INSTR(job_title,'M',   1, 1) AS A
     , INSTR(job_title,'M',   1, 2) AS B
     , INSTR(job_title,'M',   2, 1) AS C
     , INSTR(job_title,'ark', 1, 1) AS D
     , INSTR(job_title,'ark', 1, 2) AS E
FROM jobs
WHERE job_id = 'MK_MAN';

JOB_TITLE                 A   B   C   D   E
------------------- --- --- --- --- ---
Marketing Manager         1  11  11   2   0
```

- List all employee last names than contain an embedded blank

```
SELECT employee_id, first_name, last_name
FROM employees
WHERE INSTR(last_name,' ') > 1;

EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- ---------------
        102 Lex                  De Haan
```

Character Functions: `LENGTH`

- The `LENGTH` functions return the length of strings
  - Implicitly converts any datatype to a string if necessary
    - Can use to determine the "length" of a number
    - Or the "length" of a date datatype in default format
- Syntax:

  `LENGTH(string)`
- Example: find the length of the email address for employee ID 102

```
SELECT first_name, last_name, email, LENGTH(email)
FROM employees
WHERE employee_id = 102;

FIRST_NAME           LAST_NAME            EMAIL           LENGTH(EMAIL)
-------------------- -------------------- --------------- -------------
Lex                  De Haan              LDEHAAN                     7
```

TO_CHAR: A Formatting Function

- The formatting function is TO_CHAR
  - It takes two parameters
    - The data to be formatted
    - The format mask to be used
- Example:

```
SELECT salary,
       TO_CHAR(salary,'$99,999')          AS sal,
       TO_CHAR(hire_date, 'Mon DD, YYYY') AS hired
FROM employees;

    SALARY SAL      HIRED
---------- -------- ------------
     24000  $24,000 Mar 10, 2002
```

Exercise 8.1: Using Scalar Functions

HANDS-ON EXERCISE

60 min

- Please complete this exercise in your Exercise Manual

Chapter Concepts

Date and Time Functions Available

- The date and time functions are:

ADD_MONTHS	LOCALTIMESTAMP	SESSIONTIMEZONE	TO_DSINTERVAL
CURRENT_DATE	MONTHS_BETWEEN	SYS_EXTRACT_UTC	TO_YMINTERVAL
CURRENT_TIMESTAMP	NEW_TIME	SYSDATE	TRUNC (date)
DBTIMEZONE	NEXT_DAY	SYSTIMESTAMP	TZ_OFFSET
EXTRACT(datetime)	NUMTODSINTERVAL	TO_CHAR(datetime)	
FROM_TZ	NUMTOYMINTERVAL	TO_TIMESTAMP	
LAST_DAY	ROUND(date)	TO_TIMESTAMP_TZ	

- Most operate on all datetime data types: `DATE`, `TIMESTAMP` (and all variations), and `INTERVAL` types (not covered on this course)
- Exceptions:
  - Only `DATE`: `ADD_MONTHS`, `CURRENT_DATE`, `LAST_DAY`, `NEW_TIME`, `NEXT_DAY`
    - If you provide a timestamp, it is converted to a `DATE` value and a `DATE` is returned
  - `MONTHS_BETWEEN` returns a number
  - `ROUND` and `TRUNC` do not accept timestamp or interval values at all

Date and Time Functions: SYSDATE and SYSTIMESTAMP

- SYSDATE is a function call that returns a date datatype
  - The setting on the server where the Oracle database resides
  - To determine the date:

```
SELECT SYSDATE FROM DUAL;

SYSDATE
-----------
13-MAY-2019
```

- SYSTIMESTAMP is a function call similar to SYSDATE
  - Returns TIMESTAMP WITH TIMEZONE from the server

```
SELECT SYSTIMESTAMP
   , TO_CHAR(SYSTIMESTAMP,'YYYY MM DD HH24 MI SS.FF') FROM DUAL;

SYSTIMESTAMP                        TO_CHAR(SYSTIMESTAMP,'YYYYMMD
----------------------------------- -----------------------------
13-MAY-19 17.39.59.086000000 +01:00 2019 05 13 17 39 59.086000
```

Date and Time Functions: `ADD_MONTHS`

- `ADD_MONTHS` returns the date `input_date` plus `integer` months
- Syntax: `ADD_MONTHS(input_date, integer)`
  - The return type is always `DATE`, regardless of the datatype of `input_date`
- If `input_date` is the last day of the month or if the resulting month has fewer days than the day component of `input_date`, then the result is the last day of the resulting month
  - Example:

```
SELECT SYSDATE,
       ADD_MONTHS(SYSDATE,2)        AS A,
       SYSDATE + 18                 AS B,
       ADD_MONTHS(SYSDATE + 18, 1)  AS C,
       ADD_MONTHS(SYSDATE + 18, -6) AS D
FROM DUAL;

SYSDATE     A           B           C           D
----------- ----------- ----------- ----------- -----------
13-MAY-2019 13-JUL-2019 31-MAY-2019 30-JUN-2019 30-NOV-2018
```

Chapter Concepts

Miscellaneous Single-Row Functions

- The following single-row functions do not fall into any of the other single-row function categories

Non-XML Functions		XML Functions	
BFILENAME	NVL	APPENDCHILDXML	XMLCDATA
COALESCE	NVL2	DELETEXML	XMLCOLATTVAL
CV	ORA_HASH	DEPTH	XMLCOMMENT
DECODE	PRESENTNNV	EXTRACT (XML)	XMLCONCAT
DUMP	PRESENTV	EXISTSNODE	XMLFOREST
EMPTY_BLOB	PREVIOUS	EXTRACTVALUE	XMLPARSE
EMPTY_CLOB	SYS_CONNECT_BY_PATH	INSERTCHILDXML	XMLPI
EXISTSNODE	SYS_CONTEXT	INSERTXMLBEFORE	XMLQUERY
GREATEST	SYS_EXTRACT_UTC	PATH	XMLROOT
LEAST	SYS_GUID	SYS_DBURIGEN	XMLSEQUENCE
LNNVL	SYS_TYPEID	SYS_XMLAGG	XMLSERIALIZE
NLS_CHARSET_DECL_LEN	UID	SYS_XMLGEN	XMLTABLE
NLS_CHARSET_ID	USER	UPDATEXML	XMLTRANSFORM
NLS_CHARSET_NAME	USERENV	XMLAGG	
NULLIF	VSIZE		

Dealing with NULLs: COALESCE

- COALESCE returns the first non-null expr in the expression list
  - If all occurrences of expr evaluate to null, then the function returns null
  - COALESCE is ANSI standard
- Syntax: COALESCE(expr1, expr2, … exprN)

```
SELECT COALESCE(NULL, 2, 3, 4)     AS A,
       COALESCE(1, NULL, 3, 4)     AS B,
       COALESCE(NULL, NULL, 3, 4) AS C
FROM DUAL;

 A          B          C
---------- ---------- ----------
         2          1          3


SELECT commission_pct, last_name, COALESCE(TO_CHAR(commission_pct), 'No Commission')
FROM employees
WHERE employee_id = 100;

COMMISSION_PCT LAST_NAME                COALESCE(TO_CHAR(COMMISSION_PCT),'NOCOMM
-------------- ------------------------ ----------------------------------------
               King                     No Commission
```

Dealing with `NULL`s: `NVL`, `NVL2`

- `NVL`, which predates the ANSI `COALESCE`, is commonly used
- Syntax: `NVL(expr1, expr2)`
  - Returns `expr1` if it is not `NULL`
  - Returns `expr2` otherwise
- Other database vendors have also implemented the `NVL` function
  - Oracle recommends using `COALESCE`
- `NVL2` is similar
  - `NVL2(expr1, expr2, expr3)`
  - Returns `expr2` if `expr1` is `NOT NULL`, returns `expr3` if it is

NVL Example

```
SELECT ename, sal, comm, sal + comm "Total Compensation"
FROM emp;

ENAME              SAL       COMM Total Compensation
---------- ---------- ---------- ------------------
SMITH              800
ALLEN             1600        300               1900
WARD              1250        500               1750
JONES             2975


SELECT ename, sal, comm, sal + NVL(comm, 0) "Total Compensation"
FROM emp;

ENAME              SAL       COMM Total Compensation
---------- ---------- ---------- ------------------
SMITH              800                           800
ALLEN             1600        300               1900
WARD              1250        500               1750
JONES             2975                          2975
```

Exercise 8.2: Additional SQL Functions

HANDS-ON EXERCISE

30 min

- Please complete this exercise in your Exercise Manual

Chapter Concepts

Basic Server Datatypes

Introduction to Functions

Scalar Functions

`DATE` Functions

Miscellaneous Functions

Chapter Summary

Chapter Summary

In this chapter, we have discussed:

- Defining the common datatypes
- Using the simple SQL functions
    - Definition
    - Classes of functions
    - Common Single Row (Scalar) Functions
- Using `DATE`-related function
- Miscellaneous functions