Chapter 5: SQL Functions

SQL Functions Overview

Functions in SQL:

- May accept **parameters** (some take none)
- Perform a process/calculation using those parameters
- Return **a single value**

Example

```sql
SELECT LASTNAME, INITCAP(LASTNAME) FROM ONLINE_SUBSCRIBERS;
```

- `INITCAP` capitalizes the first letter of each word, makes the rest lowercase
- Note: It's not perfect (e.g., McLean becomes Mclean), so use appropriately

Functions are **called/invoked** from within SQL statements:

- SELECT, INSERT, UPDATE, DELETE clauses, etc.

Function Types

1. **Built-in Functions**: Provided by SQL; always available
2. **User-defined Functions**: Custom-coded using languages like PL/SQL (*Not tested on the exam*)

Function Categories

1. Character Functions (manipulate text)

- `LENGTH`, `RPAD`, `LPAD`, `RTRIM`, `LTRIM`, `TRIM`, `INSTR`, `SUBSTR`, `REPLACE`, `SOUNDEX`, etc.

2. Number Functions (math and numeric logic)

- Trig: `SIN`, `COS`, `TAN`, etc.
- Others: `ABS`, `SIGN`, `ROUND`, `TRUNC`
- Can be combined with operators: `+`, `-`, `*`, `/`

3. Date Functions (handle date/time values)

- Current date/time: `SYSDATE`, `SYSTIMESTAMP`
- Rounding: `ROUND`, `TRUNC`
- Differences: simple subtraction or `MONTHS_BETWEEN`
- Date arithmetic: `ADD_MONTHS`
- Example use: check if a date is on a Saturday (via conversion functions)

4. Other Functions

- Functions that don't fit the above categories
- Example: `USER` - returns current user name (takes no parameters)

The DUAL Table

- One-row, one-column (DUMMY = 'X') table used to run functions or expressions without querying actual tables
- Example:

```sql
SELECT SYSDATE FROM DUAL;
```

Character Functions Detail

UPPER / LOWER

- **Syntax:** `UPPER(s1)`, `LOWER(s1)`
- **Process:** Converts string to uppercase or lowercase
- **Use:** Normalize case for comparisons
- **Example:**

```sql
SELECT EMPLOYEE_ID FROM EMPLOYEES
WHERE UPPER(LAST_NAME) = 'MCGILLICUTTY';
```

INITCAP

- **Syntax:** `INITCAP(s1)`
- **Process:** Capitalizes the first letter of each word
- **Note:** May mishandle special characters (e.g., 'McDonald''s' → 'Mcdonald'S')

- **Example:**

```sql
SELECT INITCAP('napoleon') FROM DUAL;
```

Single Quote Escape

Use two single quotes to escape one:

```sql
SELECT 'O''Hearn' FROM DUAL;
```

CONCAT and ||

- **Syntax:** `CONCAT(s1, s2)` or `s1 || s2`
- **Process:** Joins two or more strings
- **Note:** `CONCAT` allows 2 params only, `||` allows unlimited
- **Example:**

```sql
SELECT 'Hello, ' || 'world!' FROM DUAL;
```

LPAD / RPAD

- **Syntax:** `LPAD(s1, n, s2)` / `RPAD(s1, n, s2)`
- **Process:** Pads s1 to length n with s2 (left or right)
- **Example:**

```sql
SELECT RPAD('Chapter One - I Am Born', 40, '.') FROM DUAL;
```

LTRIM / RTRIM

- **Syntax:** `LTRIM(s1, s2)` / `RTRIM(s1, s2)`
- **Process:** Removes s2 from the start or end of s1
- **Default:** If s2 not given, trims spaces
- **Example:**

```sql
SELECT RTRIM('Seven thousand-------', '-') Result FROM DUAL;
```

TRIM

- **Syntax:** `TRIM([LEADING | TRAILING | BOTH] trim_char FROM trim_source)`
- **Defaults:** If trim_char or direction not given, defaults to space and BOTH
- **Example:**

```sql
SELECT TRIM(TRAILING '-' FROM 'Seven thousand-------') FROM DUAL;
```

LENGTH

- **Syntax:** `LENGTH(s)`
- **Output:** Numeric; returns string length
- **Example:**

```sql
SELECT LENGTH('Supercalifragilisticexpialidocious') FROM DUAL;
```

INSTR

- **Syntax:** `INSTR(s1, s2, pos, n)`
- **Finds:** nth occurrence of s2 in s1 starting at pos
- **Negative pos:** Searches backward from end
- **Example:**

```sql
SELECT INSTR('Mississippi', 'is', 1, 2) FROM DUAL;
```

SUBSTR

- **Syntax:** `SUBSTR(s, pos, len)`
- **Extracts:** len characters from s starting at pos
- **If len omitted:** Goes to end of string

- **Negative pos:** Counts from end
- **Example:**

```sql
SELECT SUBSTR('Name: MARK KENNEDY', 7) FROM DUAL;
```

SOUNDEX

- **Syntax:** `SOUNDEX(s)`
- **Purpose:** Encodes string phonetically
- **Rules:**
  - First letter stays the same
  - Convert next letters to codes:

Letters	Code
B, F, P, V	1
C, G, J, K...	2
D, T	3
L	4
M, N	5
R	6
A, E, H, I...	Ignored

- **Example:**

```sql
SELECT SOUNDEX('Worthington'), SOUNDEX('Worthen') FROM DUAL;
-- Both return W635
```

- **Use in WHERE:**

```sql
SELECT EMPLOYEE_ID FROM EMPLOYEES
WHERE SOUNDEX(LAST_NAME) = SOUNDEX('Worthen');
```

- **Note:** Best with English words; inconsistent with non-English names

Numerical Functions

CEIL(n)

- Returns the **smallest integer** ≥ n

FLOOR(n)

- Returns the **largest integer** ≤ n

ROUND(n, i)

- Rounds n to i decimal places
  - i > 0 → decimal right
  - i = 0 → nearest whole number
  - i < 0 → digit left of decimal
  - Ties round **away from zero**
- Output type:
  - Omit i: same type as n
  - Include i: returns NUMBER

TRUNC(n, i)

- Cuts off digits beyond i decimal places (no rounding)
- i rules same as ROUND
- Always rounds **toward zero**

REMAINDER(n1, n2)

- Returns difference between n1 and the **nearest multiple** of n2
- May return **negative** if nearest multiple is above n1

MOD(n1, n2)

- Same as REMAINDER, but uses **FLOOR** instead of ROUND
- Always returns **expected modulus** (positive if n1 > n2)

Date Functions

SYSDATE

- Returns current **system date and time** from the DB server

- Output includes **time**, though not visible unless formatted with `TO_CHAR`

ROUND(d, i)

- Rounds date d to the nearest unit:
  - 'DD' → day (default if i omitted)
  - 'MM', 'YYYY', etc. → month, year
  - Noon or later rounds **up**

TRUNC(d, i)

- Same as ROUND but always rounds **down**
- Uses same format models as ROUND

NEXT_DAY(d, c)

- Returns first occurrence of day c **after** date d
- c is text (e.g., 'Monday', 'Sat')

LAST_DAY(d)

- Returns the **last day of the month** in which date d falls

ADD_MONTHS(d, n)

- Adds n months to date d
- To subtract months, use **negative n**

MONTHS_BETWEEN(d1, d2)

- Returns the **number of months** between d1 and d2
- Can return **decimal values** for partial months
- Positive if d1 > d2, negative otherwise
- Watch parameter order for correct sign

NUMTOYMINTERVAL(n, interval_unit)

- Converts a number to an `INTERVAL YEAR TO MONTH`
- interval_unit = 'YEAR' or 'MONTH'
- Example:

```sql
SELECT NUMTOYMINTERVAL(27, 'MONTH') FROM DUAL;
-- Outputs: 2-3 (2 years, 3 months)
```

NUMTODSINTERVAL(n, interval_unit)

- Converts number to `INTERVAL DAY TO SECOND`
- interval_unit = 'DAY', 'HOUR', 'MINUTE', 'SECOND'
- Example:

```sql
SELECT NUMTODSINTERVAL(36, 'HOUR') FROM DUAL;
-- Outputs: 1 12:0:0.0 (1 day, 12 hours)
```

Date Arithmetic with Numbers

- 1 = 1 day
- 1 hour = 1/24
- 1 minute = 1/1440
- Example: `SYSDATE + 1/1440` → Adds 1 minute

Function Categories by Row Processing

Scalar Functions

- 1 input row → 1 output (e.g., `UPPER`)

Aggregate Functions

- Many input rows → 1 result (e.g., `SUM`)

Analytical Functions

- Works on "windows" of rows using `OVER()`
- **Only allowed in SELECT or ORDER BY** clauses
- **Not allowed** in WHERE, GROUP BY, HAVING

Analytical Functions

OVER, PARTITION BY, ORDER BY

- `SUM(...) OVER (ORDER BY ...)` → running total
- `ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING` → sliding window
- `PARTITION BY` → resets window per group
- Changing `ORDER BY` in `SELECT` does **not** change analytic function output

LAG(column), LEAD(column)

- Returns previous or next row's value
- Can specify offset (default: 1). `LAG(col, 2)` → value 2 rows before
- Uses `OVER (ORDER BY ...)`
- Null if no row exists at offset

STDDEV & VARIANCE

- `STDDEV()` = standard deviation of sample
- `VARIANCE()` = variance of sample
- Example:

```sql
SELECT STDDEV(SQ_FT), VARIANCE(SQ_FT)
FROM SHIP_CABINS;
```

- Can be used with `OVER(...)` for cumulative values

PERCENTILE_CONT(p)

- Calculates a percentile using linear interpolation
- Syntax:

```sql
PERCENTILE_CONT(0.6) WITHIN GROUP (ORDER BY col)
OVER (PARTITION BY group_col)
```

- Finds target row using: TR = 1 + (P * (n - 1))
- Interpolated using CEIL, FLOOR, and their values

Nesting Functions

Overview

- **Nested function**: A function whose output is used as input to another function
- The **innermost (inner)** function executes **first**, followed by the **outer** function
- You can nest **multiple functions at multiple levels**
- This section covers **only scalar functions** (single-row functions), not aggregate functions

Example: SUBSTR and INSTR Combination

Goal: Extract the **state abbreviation** from addresses in ADDRESS2, where the pattern is: `<City>, <State Abbreviation> <ZIP>`

Key functions:

- `INSTR(ADDRESS2, ',')` finds the position of the comma
- Add **2** to get past the comma and space
- Nest in `SUBSTR` to extract the two-letter state code

Query:

```sql
SELECT
    ADDRESS2,
    INSTR(ADDRESS2, ',') THE_COMMA,
    SUBSTR(ADDRESS2, INSTR(ADDRESS2, ',') + 2, 2) STATE
FROM
    ORDER_ADDRESSES
ORDER BY
    3;
```

- `ORDER BY 3` sorts by extracted state abbreviation
- This shows how nested functions can extract and sort based on dynamic positions when there's a consistent pattern in data