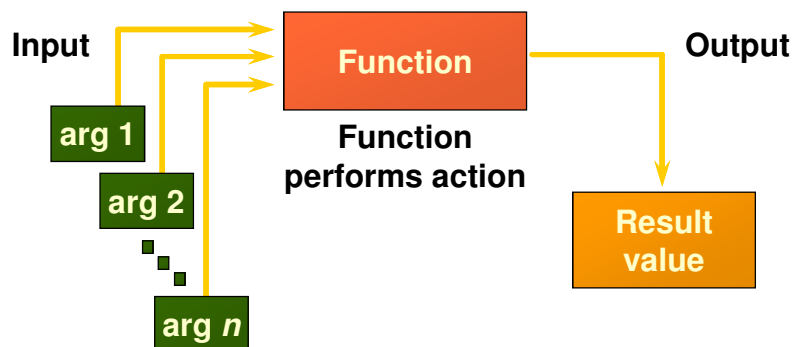


Chapter 4

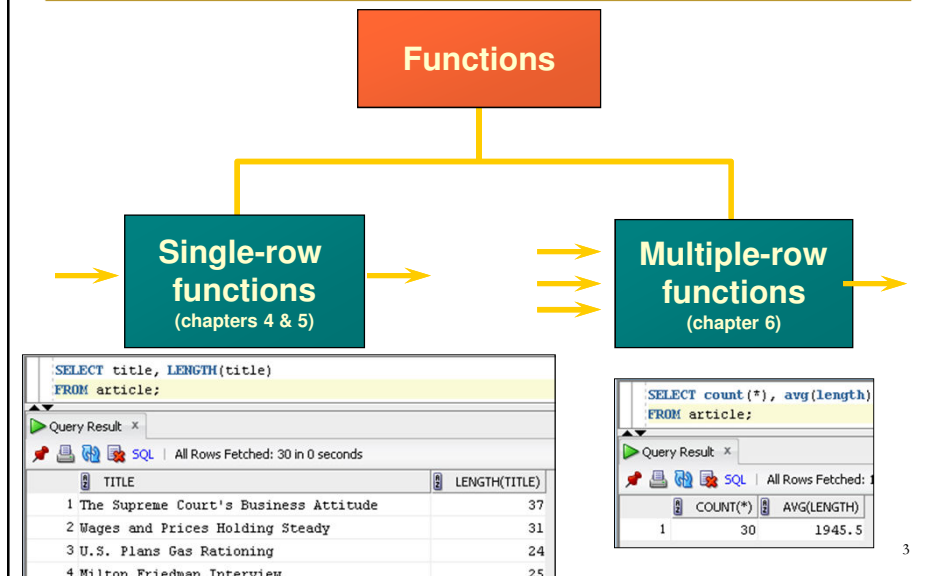
Character, Number, and Miscellaneous Functions

SQL Functions

- A block of executable program code that can accept arguments, performs processing, and returns a result



Two Types of SQL Functions



`function_name([arg1, arg2, ...])`

Single-Row Functions

- Manipulate data
- Accept arguments and return a value
- Act on each row
- Returns one result per row
- May modify the datatype
- May be nested

Character Case Conversion Functions

Function	Result
LOWER(' SQL Course ')	
UPPER(' SQL Course ')	
INITCAP(' SQL Course ')	

■ Practice Time

```
SELECT ln, fn FROM writer WHERE ln = 'HALL';  
SELECT ln, fn FROM writer WHERE UPPER(ln) = 'HALL';
```

5

Character Manipulation Functions

Function	Result
LPAD(5000,10,'*')	
RPAD(5000,10,'+')	
LENGTH('String')	
SUBSTR('Tomorrow', 3, 4)	
SUBSTR('Tomorrow', 5)	
INSTR('String', 'r')	
INSTR('Mississippi River', 'i', 3, 4)	
CONCAT('Good', 'String')	

6

Practice Time: Character Functions

```
SELECT LPAD(ln, 20, ' ') FROM writer;

SELECT RPAD(ln, 20, '.') FROM writer;

SELECT title, LENGTH(title)
FROM article
ORDER BY 2 DESC;

SELECT phone, SUBSTR(phone, 2, 3) Prefix FROM writer;

SELECT title, issue
FROM article
WHERE INSTR(issue, 'MAY') <> 0;

SELECT cardnum, SUBSTR(cardnum, -4) AS last4,
expires, limit
FROM ttrollen.creditcard;
```

7

Character Manipulation Functions

Function	Result
LTRIM(' Tom')	
RTRIM('Tom ')	
RTRIM('852220000', '0')	
TRIM(' Scottsdale ')	
REPLACE(street_address, 'St.', 'Street')	

```
SELECT street_address
FROM student
WHERE INSTR(street_address, 'St.') > 0;

SELECT street_address,
REPLACE(street_address, 'St.', 'Street')
FROM student
WHERE INSTR(street_address, 'St.') > 0;
```

8

Concatenation Operator ||

- Combines two character strings into one result string

```
SELECT last_name || ', ' || first_name, phone  
FROM instructor;
```

```
SELECT last_name || ', ' || first_name AS INSTRUCTOR, phone  
FROM instructor;
```

```
SELECT city || ', ' || state || ' ' || zip LOCALE  
FROM zipcode;
```

- VS CONCAT ()

9

Dual

- A **pseudotable** that permits a SELECT statement to return the result of a function without using an actual table's data

```
DESCR DUAL
```

```
SELECT * FROM DUAL;
```

```
SELECT USER, SYSDATE FROM DUAL;
```

```
SELECT 3*7/12 FROM DUAL;
```

10

Nested Functions

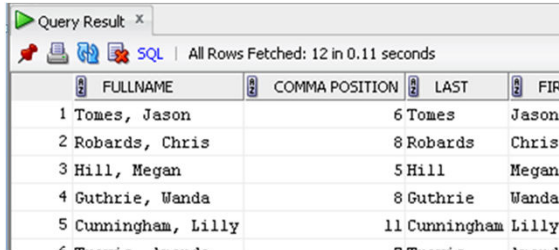
- Are evaluated from innermost outward

- watch the pairs of parentheses!

```
SELECT ln, fn,  
       LOWER(CONCAT(SUBSTR(fn,1,1), SUBSTR(ln,1,1))) Initials  
FROM writer;
```

- Practice Time

- schema `ttrollen` has a table named `maillist`
- view the table's structure
- view the table's data
- produce the following
 - incremental development



	FULLNAME	COMMA POSITION	LAST	FIR
1	Tomes, Jason		6 Tomes	Jason
2	Robards, Chris		8 Robards	Chris
3	Hill, Megan		5 Hill	Megan
4	Guthrie, Wanda		8 Guthrie	Wanda
5	Cunningham, Lilly		11 Cunningham	Lilly

```
TRANSLATE(input_expression, from_string, to_string)
```

TRANSLATE Function

- Returns a string with all occurrences of each character in `from_string` replaced by corresponding character in `to_string`
 - characters not present in `from_string` are not replaced

- Eg: Translate a title into a string that could be used as a filename

```
SELECT TRANSLATE('SQL*Plus: Reference/Style Guide <draft3?>',  
                '/\:*?"<>', '_____') as filename  
FROM DUAL;
```

- WriterID's should follow a pattern of LDDD

```
SELECT writerid, ln, fn,  
       TRANSLATE(writerid,  
                 '0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZ',  
                 'DDDDDDDDDDLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL')  
FROM writer;
```

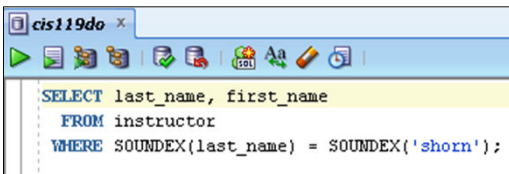
- how show only rows that violate the LDDD pattern?

12

SOUNDEX(*string_expression*)

Soundex Function

- Returns a string containing the phonetic representation of the input string
 - compare words that are spelled differently, but sound alike in English
- Practice Time
 - Caller says her last name but you're not sure of spelling



```
SELECT last_name, first_name
FROM instructor
WHERE SOUNDEX(last_name) = SOUNDEX('shorn');
```

13

Lab 4.2

Number Functions

Function	Result
ABS(-20)	
MOD(90, 60)	
ROUND(63.7352, 2)	
TRUNC(63.7352, 2)	
ROUND(63.7352, -1)	
ROUND(63.7352, -2)	
FLOOR(63.7352)	
CEIL(63.7352)	
CEIL(63.0001)	

14

Arithmetic Operators

Operator	Meaning
*	multiplication
/	division
+	addition
-	subtraction

- Shown in order of **precedence**
- **Practice Time**

```
SELECT 3+4*5 FROM DUAL;  
SELECT (3+4)*5 FROM DUAL;  
SELECT ln, amount, amount + 50  
FROM writer  
WHERE amount <> 0;
```

15

Practice Time

1. Alex filled up his SUV with 21.3 gallons at odometer reading 23,557. His previous fill-up was at odometer reading 23,266. How many MPG did he get?
2. Each article is to be rewritten to condense it to 65% of its original length. Show each title, original length and new length. Round calculated new lengths to the closest whole number. Entitle the column **Goal**.
3. Using string functions, show the title of any article containing **War**.
4. A PowerPoint presentation has 22 slides. When printed 3-to-a-page, how many pages are needed and how many slides will appear on the last page?

16

Nulls

- Nulls cause problems in calculations
 - the result will be Null if any of its inputs are Null

The left screenshot shows the following SQL query and its results:

```
DESCRIBE scott.emp;
SELECT ename, sal, comm,
       sal + comm AS Total
FROM scott.emp;
```

ENAME	SAL	COMM	TOTAL
1 SMITH	800	(null)	(null)
2 ALLEN	1600	300	1900
3 WARD	(null)	500	(null)
4 JONES	2975	(null)	(null)
5 MARTIN	1250	1400	2650

The right screenshot shows the same data with the NVL function used to replace null values with zero:

```
SELECT ename, sal, comm,
       NVL(sal,0) + NVL(comm,0) AS Total
FROM scott.emp;
```

ENAME	SAL	COMM	TOTAL
1 SMITH	800	(null)	800
2 ALLEN	1600	300	1900
3 WARD	(null)	500	500
4 JONES	2975	(null)	2975
5 MARTIN	1250	1400	2650
6 BLAKE	2850	(null)	2850
7 CLARK	2450	(null)	2450
8 SCOTT	3000	(null)	3000

`NVL(input_expression, substitute_expression)`

NVL Function

- Replaces a Null with a substitute value
 - if input_expression has a value, NVL returns input_expression
 - if input_expression is Null, NVL returns substitute_expression
- Datatypes must be compatible
 - NVL(phone, 'Not on file') --both are text
 - NVL(amount, 0) --both are numeric
 - NVL(hiredate, '01-JAN-97') --both are dates

Practice Time

```
SELECT ln, fn, NVL(contact, 'no supervisor') Boss
FROM writer;
```

`NVL2(input_expr, not_null_subst_expr, null_subst_expr)`

NVL2 Function

- Returns one of two values, depending on whether input_expr is NULL
 - if input_expr has a value, NVL2 returns not_null_subst_expr
 - if input_expr is Null, NVL2 returns null_subst_expr
- Datatypes of last 2 arguments must be compatible

Practice Time

```
SELECT ln, fn,  
       NVL2(phone, 'On file', 'Not on file') "Phone Status"  
FROM writer;
```

```
SELECT ln, fn,  
       NVL2(phone, phone, 'Not on file') "Phone Status"  
FROM writer;
```

19

`COALESCE(input_expr, subst_expr1 [, subst_expr2], [...])`

COALESCE Function

- Searches list of replacement values for the first non-null value available
 - returns Null if all the expressions are Null

- Datatypes must be compatible

Practice Time

```
SELECT last_name, first_name,  
       COALESCE(work_phone, cell_phone, home_phone)  
FROM employee;
```

20

```
NULLIF(input_expr1, input_expr2)
```

NULLIF Function "null if equal"

- Used to compare two values and display a result
 - if input_expr1 = input_expr2, NULLIF returns Null
 - if input_expr1 ≠ input_expr2, NULLIF returns input_expr1

- Practice Time

```
SELECT student_id, last_name, created_date, modified_date,  
       NULLIF(created_date, modified_date)  
FROM student;
```

```
SELECT student_id, last_name, created_date, modified_date  
FROM student  
WHERE NULLIF(created_date, modified_date) IS NOT NULL;
```

21

DECODE Function

```
DECODE(expression, equals_search1, then_result1  
        [, equals_search2, then_result2]  
        [, ...]  
        [, else_default])
```

- Provides IF...THEN...ELSE logic to display substitute values
- Practice Time

```
SELECT ln, freelancer,  
       DECODE(freelancer, 'Y', 'Freelancer',  
              'N', 'Staff Writer',  
              NULL, 'missing',  
              'Invalid code') AS Status  
FROM writer;
```

22

Searched CASE Expression

```
CASE {WHEN condition THEN return_expr  
      [WHEN condition THEN return_expr] ...}  
      [ELSE else_expr]  
END
```

- Specifies a set of alternate return values
- Conditions
 - are evaluated sequentially; 1st true one is returned
 - can include ranges, Boolean operators
- Can be used in SELECT, WHERE, or ORDER BY clauses
- Datatypes of the return values must be compatible

```
SELECT ename, sal, CASE WHEN sal <1000 THEN 'grunt'  
                        WHEN sal <=3000 THEN 'drone'  
                        WHEN sal IS NULL THEN 'unknown'  
                        ELSE 'honcho'  
                        END AS Status  
FROM scott.emp;
```

23

Simple CASE Expression

```
CASE {expr WHEN comparison_expr THEN return_expr  
      [WHEN comparison_expr THEN return_expr]...}  
      [ELSE else_expr]  
END
```

- When all conditions involve equalities
 - are evaluated sequentially; 1st true one is returned
- Datatypes of the return values must be compatible

```
SELECT ln, fn, amount,  
       CASE freelancer WHEN 'Y' THEN amount + 250  
                    WHEN 'N' THEN amount + 100  
       END AS "Proposed Bonus"  
FROM writer;
```

24