EXPERIMENT - VI
STRING FUNCTIONS AND PATTERN MATCHING

August 10, 2019

ADITHYA D RAJAGOPAL
ROLL NO : 9
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

AIM

To study the following string functions and pattern matching operators:

1. SUBSTR
2. RPAD
3. INITCAP
4. INSTR
5. LPAD
6. CONCAT
7. LTRIM
8. UPPER
9. LENGTH
10. RTRIM
11. LOWER
12. REVERSE

STRING FUNCTIONS

The main string functions in SQL are as follows:

SUBSTR(field name/string,n,m)

Returns a portion of char, beginning at character m, n characters long.

1. If m is 0, it is treated as 1.
2. If m is positive, Oracle counts from the beginning of char to find the first character.
3. If m is negative, Oracle counts backwards from the end of char.
4. If n is omitted, Oracle returns all characters to the end of char. If n is less than 1, a null is returned.

Floating-point numbers passed as arguments to SUBSTR are automatically converted to integers.

RPAD(field name/string,length,character)

Returns char1, right-padded to length n with char2, replicated as many times as necessary; char2 defaults to a single blank. If char1 is longer than n, this function returns the portion of char1 that fits in n.
The argument n is the total length of the return value as it is displayed on your terminal screen. In most character sets, this is also the number of characters in the return value. However, in some multibyte character sets, the display length of a character string can differ from the number of characters in the string.

INITCAP(field name/string)

The Oracle INITCAP() function sets the first letter of each word in uppercase, all other letters in lowercase. Words are delimited by white space or characters that are not alphanumeric. A string whose first character in each word will be converted to uppercase and the rest characters will be converted to lowercase.

INSTR(field name/string,substring,n,m)

Searches char1 beginning with its nth character for the nth occurrence of char2 and returns the position of the character in char1 that is the first character of this occurrence. If n is negative, Oracle counts and searches backward from the end of char1. The value of m must be positive. The default values of both n and m are 1, meaning Oracle begins searching at the first character of char1 for the first occurrence of char2. The return value is relative to the beginning of char1, regardless of the value of n, and is expressed in characters. If the search is unsuccessful (if char2 does not appear m times after the nth character of char1) the return value is 0.

LPAD(field name/string,length,character)

Returns char1, left-padded to length n with the sequence of characters in char2; char2 defaults to a single blank. If char1 is longer than n, this function returns the portion of char1 that fits in n.
The argument n is the total length of the return value as it is displayed on your terminal screen. In most character sets, this is also the number of characters in the return value. However, in some multibyte character sets, the display length of a character string can differ from the number of characters in the string.

CONCAT(field name/string, field name/string)

Combines the first and second string into single one.

LTRIM(field name/string,substring)

Returns char, with all the leftmost characters that appear in set removed; set defaults to a single blank. If char is a character literal, you must enclose it in single quotes.

UPPER(field name/string)

Gives the content in upper case letters.

LENGTH(field name/string)

Gives the length of the string.

RTRIM(field name/string,substring)

Returns char, with all the rightmost characters that appear in set removed; set defaults to a single blank. If char is a character literal, you must enclose it in single quotes.

LOWER(field name/string)

Gives the content in lowercase letters.

REVERSE(field name/string)

Gives the reverse of the string.

PATTERN MATCHING OPERATORS

LIKE Operator

The LIKE operator is used in character string comparisons with pattern matching.
With the LIKE operator, you can compare a value to a pattern rather than to a constant. The pattern must appear after the LIKE keyword.
Patterns typically use special characters that Oracle matches with different characters in the value:

1. An underscore (_) in the pattern matches exactly one character (as opposed to one byte in a multibyte character set) in the value.

2. A percent sign (%) in the pattern can match zero or more characters (as opposed to bytes in a multibyte character set) in the value. Note that the pattern '%' cannot match a null.

Case Sensitivity and Pattern Matching

Case is significant in all conditions comparing character expressions including the LIKE and equality (=) operators. You can use the UPPER() function to perform a case-insensitive match.

QUESTIONS

1. Create a table named acct_details and populate the table as shown below.

Acct_No	Branch	Name	Phone
A40123401	Chicago	Mike Adams	(378) 400-1234
A40123402	Miami	Diana George	(372) 420-2345
B40123403	Miami	Diaz Elizabeth	(371) 450-3456
B40123404	Atlanta	Jeoffrey George	(370) 460-4567
B40123405	New York	Jennifer Kaitlyn	(373) 470-5678
C40123406	Chicago	Kaitlyn Vincent	(318) 200-3235
C40123407	Miami	Abraham Gottfield	(328) 300-2256
C50123408	New Jersey	Stacy Williams	(338) 400-5237
D50123409	New York	Catherine George	(348) 500-6228
D50123410	Miami	Oliver Scott	(358) 600-7230

```
postgres=# SELECT * FROM ACCT_DETAILS;
  acc_no   |   branch   |       name         |     phone
-----------+------------+--------------------+---------------
 A40123401 | Chicago    | Mike Adams         | (378)400-1234
 A40123402 | Miami      | Diana George       | (372)420-2345
 B40123403 | Miami      | Diaz Elizabeth     | (371)450-3456
 B40123404 | Atlanta    | Jeoffrey George    | (370)460-4567
 B40123405 | New York   | Jennifer Kaitlyn   | (373)470-5678
 C40123406 | Chicago    | Kaitlyn Vincent    | (318)200-3235
 C40123407 | Miami      | Abraham Gottfield  | (328)300-2256
 C50123408 | New Jersey | Stacy Williams     | (338)400-5237
 D50123409 | New York   | Catherine George   | (348)500-6228
 D50123410 | Miami      | Oliver Scott       | (358)600-7230
(10 rows)

postgres=#
```

(a) Find the names of all people starting on the alphabet 'D'

```
postgres=# SELECT NAME FROM ACCT_DETAILS
WHERE NAME LIKE 'D%';
      name
---------------
 Diana George
 Diaz Elizabeth
(2 rows)

postgres=#
```

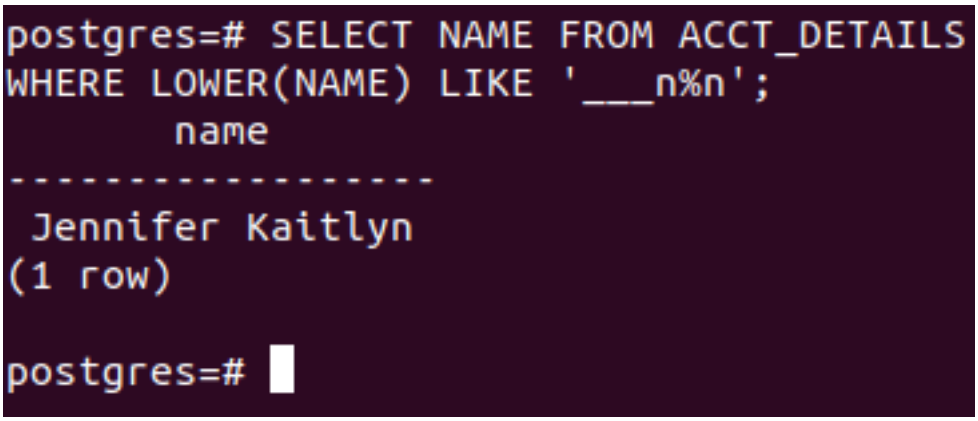
(b) List the names of all branches containing the substring 'New'

```
postgres=# SELECT BRANCH FROM ACCT_DETAILS
WHERE BRANCH LIKE 'New%';
   branch
------------
 New York
 New Jersey
 New York
(3 rows)

postgres=#
```

(c) List all the names in Upper Case Format

```
postgres=# SELECT UPPER(NAME) FROM ACCT_DETAILS;
       upper
--------------------
 MIKE ADAMS
 DIANA GEORGE
 DIAZ ELIZABETH
 JEOFFREY GEORGE
 JENNIFER KAITLYN
 KAITLYN VINCENT
 ABRAHAM GOTTFIELD
 STACY WILLIAMS
 CATHERINE GEORGE
 OLIVER SCOTT
(10 rows)

postgres=#
```

(d) List the names where the 4th letter is 'n' and last letter is 'n'



(e) List the names starting on 'D' , 3rd letter is 'a' and contains the substring 'Eli'

```
postgres=# SELECT NAME FROM ACCT_DETAILS
postgres-# WHERE NAME LIKE 'D_a%'
postgres-# AND NAME LIKE '%Eli%';
      name
---------------
 Diaz Elizabeth
(1 row)

postgres=#
```

(f) List the names of people whose account number ends in '6'

```
postgres=# SELECT NAME FROM ACCT_DETAILS
postgres-# WHERE ACC_NO LIKE '%6';
      name
-----------------
 Kaitlyn Vincent
(1 row)

postgres=#
```

(g) Update the table so that all the names are in Upper Case Format

```
postgres=# UPDATE ACCT_DETAILS
postgres-# SET NAME=UPPER(NAME);
UPDATE 10
postgres=# SELECT * FROM ACCT_DETAILS;
  acc_no   |   branch   |       name         |     phone
-----------+------------+--------------------+---------------
 A40123401 | Chicago    | MIKE ADAMS         | (378)400-1234
 A40123402 | Miami      | DIANA GEORGE       | (372)420-2345
 B40123403 | Miami      | DIAZ ELIZABETH     | (371)450-3456
 B40123404 | Atlanta    | JEOFFREY GEORGE    | (370)460-4567
 B40123405 | New York   | JENNIFER KAITLYN   | (373)470-5678
 C40123406 | Chicago    | KAITLYN VINCENT    | (318)200-3235
 C40123407 | Miami      | ABRAHAM GOTTFIELD  | (328)300-2256
 C50123408 | New Jersey | STACY WILLIAMS     | (338)400-5237
 D50123409 | New York   | CATHERINE GEORGE   | (348)500-6228
 D50123410 | Miami      | OLIVER SCOTT       | (358)600-7230
(10 rows)

postgres=#
```

(h) List the names of all people ending on the alphabet 't'

```
postgres=# SELECT NAME FROM ACCT_DETAILS
WHERE NAME LIKE '%T';
      name
-----------------
 KAITLYN VINCENT
 OLIVER SCOTT
(2 rows)

postgres=#
```

(i) List all the names in reverse

```
postgres=# SELECT REVERSE(NAME) FROM ACCT_DETAILS;
      reverse
-------------------
 SMADA EKIM
 EGROEG ANAID
 HTEBAZILE ZAID
 EGROEG YERFFOEJ
 NYLTIAK REFINNEJ
 TNECNIV NYLTIAK
 DLEIFTTOG MAHARBA
 SMAILLIW YCATS
 EGROEG ENIREHTAC
 TTOCS REVILO
(10 rows)

postgres=#
```

(j) Display all the phone numbers including US Country code (+1).
For eg: (378)400-1234 should be displayed as +1(378)400-1234. Use LPAD function.

```
postgres=# SELECT LPAD(PHONE,15,'+1')
FROM ACCT_DETAILS;
      lpad
-----------------
 +1(378)400-1234
 +1(372)420-2345
 +1(371)450-3456
 +1(370)460-4567
 +1(373)470-5678
 +1(318)200-3235
 +1(328)300-2256
 +1(338)400-5237
 +1(348)500-6228
 +1(358)600-7230
(10 rows)

postgres=#
```

(k) Display all the account numbers. The starting alphabet associated with the Account_No should be removed. Use LTRIM function.

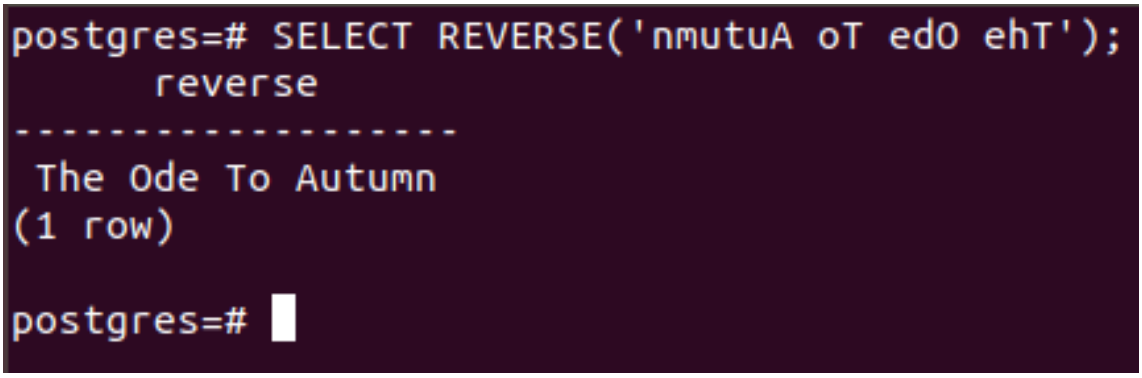
```
postgres=# UPDATE ACCT_DETAILS
postgres-# SET ACC_NO=LTRIM(ACC_NO,'ABCD');
UPDATE 10
postgres=# SELECT * FROM ACCT_DETAILS;
  acc_no  |   branch   |       name         |     phone
----------+------------+--------------------+---------------
 40123401 | Chicago    | MIKE ADAMS         | (378)400-1234
 40123402 | Miami      | DIANA GEORGE       | (372)420-2345
 40123403 | Miami      | DIAZ ELIZABETH     | (371)450-3456
 40123404 | Atlanta    | JEOFFREY GEORGE    | (370)460-4567
 40123405 | New York   | JENNIFER KAITLYN   | (373)470-5678
 40123406 | Chicago    | KAITLYN VINCENT    | (318)200-3235
 40123407 | Miami      | ABRAHAM GOTTFIELD  | (328)300-2256
 50123408 | New Jersey | STACY WILLIAMS     | (338)400-523
 50123409 | New York   | CATHERINE GEORGE   | (348)500-6228
 50123410 | Miami      | OLIVER SCOTT       | (358)600-7230
(10 rows)

postgres=#
```

(l) Display the details of all people whose account number starts in '5' and name contains the substring 'Williams'.

```
postgres=# SELECT * FROM ACCT_DETAILS
WHERE ACC_NO LIKE '5%'
AND NAME LIKE '%WILLIAMS%';
  acc_no  |   branch   |      name       |     phone
----------+------------+-----------------+---------------
 50123408 | New Jersey | STACY WILLIAMS | (338)400-523
(1 row)

postgres=#
```

2. Use the system table DUAL for the following questions:

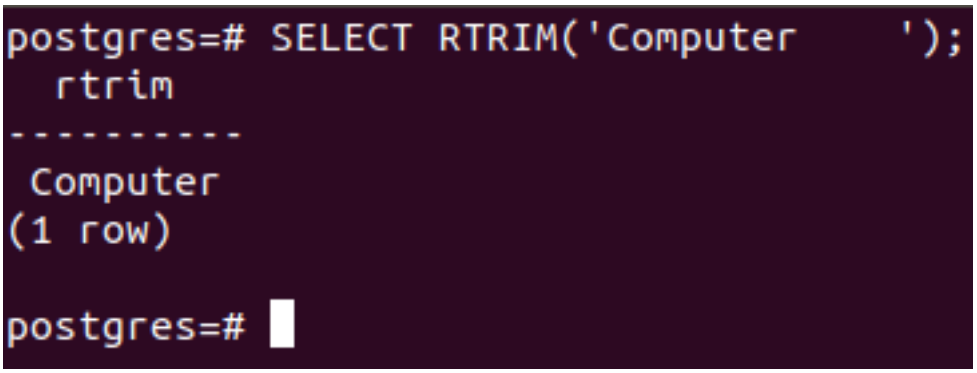
   (a) Find the reverse of the string 'nmutuAotedOehT'



(b) Use LTRIM function on '123231xyzTech' so as to obtain the output 'Tech'

```
postgres=# SELECT LTRIM('123231xyzTech','123xyz');
 ltrim
-------
 Tech
(1 row)

postgres=#
```

(c) Use RTRIM function on 'Computer ' to remove the trailing spaces.



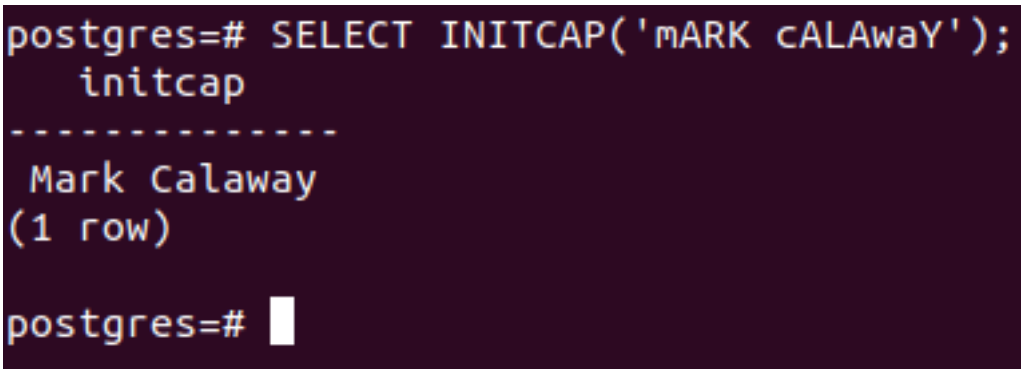
(d) Perform RPAD on 'computer' to obtain the output as 'computerXXXX'

```
postgres=# SELECT RPAD('Computer',12,'X');
     rpad
--------------
 ComputerXXXX
(1 row)

postgres=#
```

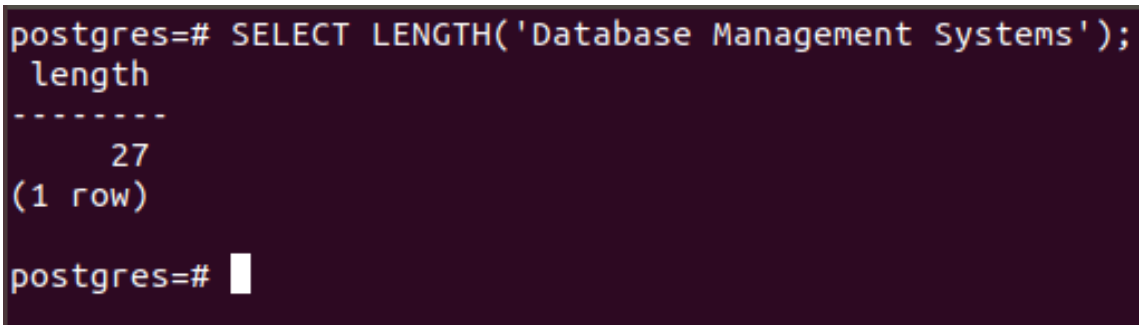
(e) Use INSTR function to find the first occurrence of 'e' in the string 'Welcome to Kerala'

```
postgres=# SELECT STRPOS('Welcome to Kerala','e');
 strpos
--------
      2
(1 row)

postgres=#
```

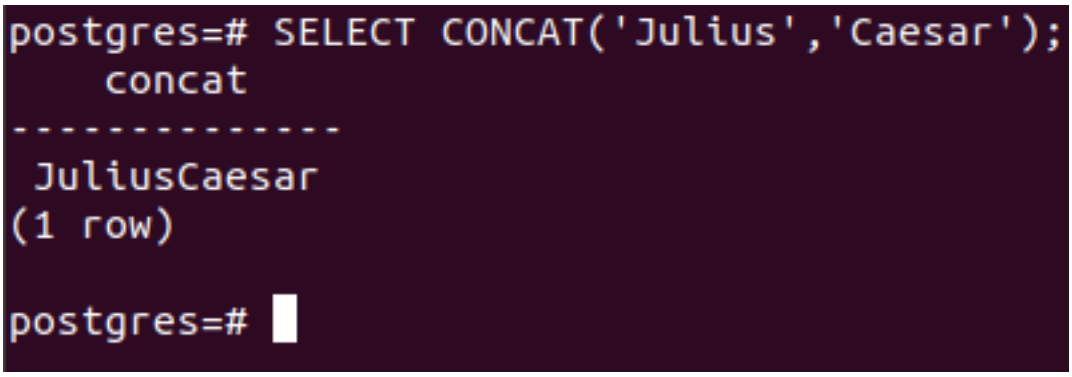
(f) Perform INITCAP function on ’mARKcALAwaY’



(g) Find the length of the string 'Database Management Systems'.



(h) Concatenate the strings 'Julius' and 'Caesar'



(i) Use SUBSTR function to retrieve the substring 'is' from the string 'India is my country'.

```
postgres=# SELECT SUBSTR('India is my country',7,2);
 substr
--------
 is
(1 row)

postgres=#
```

(j) Use INSTR function to find the second occurrence of 'k' from the last. The string is 'Making of a King'.

```
postgres=# SELECT STRPOS(REVERSE('Making of a King'),'k',2);
 strpos
--------
      0
(1 row)

postgres=#
```

RESULT

The query was executed and the output was obtained.