Table of Contents

1. SQL wildcards

Wildcard characters are used when you want to do a **partial string match**. Suppose you are looking for any product that has the word "mixer" in the product description. We might want to find 'portable mixer', '6-speed mixer'. We can use the Like operator and a wildcard pattern to find all products which contain the pattern 'mixer'- this will also find 'cement mixers' but that is the way that wildcard matching works. We cannot really do a **word** match, but we can do a pattern match.

Wildcards are used for matching strings. It is acceptable to match digits in text values- such as in an ISBN code. Oracle will do wildcard matching with numeric values but this is not a good idea- **wildcards are designed for string matching and numbers are not strings.** It is common to do wildcard matches with date values- but that relies on date formats. There are safer ways and generally more efficient ways to test dates and numeric values.

The wildcard patterns are interpreted as wildcards only if you use the LIKE operator.

You can also test using the operator NOT LIKE to find rows that do not match your pattern.

Do not use the LIKE operator if you are not using wildcards. The Like operator is more expensive- if you are testing if the customer city is Chicago- that is an equality test, not a wildcard test.

The wildcard patterns used are:

% for zero or more characters, matching any character

_ for a single character, matching any character

For these examples we are using a table named z_wildcards with the following rows:

```
    CUSTID  CUSTPHONE     CUSTADDRESS
----------  ------------  --------------------
         1  510-239-8989  101 Bush Street
         2  510-45-78785  One Bush Street
         3  415-809-8989  124 High
         4  415-124-2398  15 High Road
         5  415-239-8523  1554 Rural Highway 12
```

Demo 01: This is the SQL to create and populate the table.

```
Create table z_wildcards (cust_id number(2), cust_phone varchar2(12),
cust_address varchar2(30) );
insert into z_wildcards   values (1, '510-239-8989',  '101 Bush Street');
insert into z_wildcards   values (2, '510-45-78785',  'One Bush Street');
insert into z_wildcards   values (3, '415-809-8989',  '124 High');
insert into z_wildcards   values (4, '415-124-2398',  '15 High Road');
insert into z_wildcards   values (5, '415-239-8523',  '1554 Rural Highway 12');
```

Demo 02: We want to locate customers with an address on Bush.

```
Select cust_id, cust_address
From z_wildcards
Where cust_Address LIKE '%Bush%';
```

```
   CUST_ID CUST_ADDRESS
---------- ------------------------------
         1 101 Bush Street
         2 One Bush Street
```

Demo 03: We want to locate customers whose phone number is in the 415 area code.

```
Select cust_id, cust_phone
From z_wildcards
Where cust_phone LIKE '415%';
```

```
  CUST_ID CUST_PHONE
--------- ------------
        3 415-809-8989
        4 415-124-2398
        5 415-239-8523
```

Demo 04: We want to find customers in the 239 exchange (the middle 3 digits of their phone number). This does not work properly.

```
Select cust_id, cust_phone
From z_wildcards
Where cust_phone LIKE '%239%'
;
```

```
   CUST_ID CUST_PHONE
---------- ------------
         1 510-239-8989
         4 415-124-2398
         5 415-239-8523
```

Demo 05: We can look for the pattern '-239-'.

```
Select cust_id, cust_phone
From z_wildcards
Where cust_phone LIKE '%-239-%';
```

```
   CUST_ID CUST_PHONE
---------- ------------
         1 510-239-8989
         5 415-239-8523
```

Demo 06: Text strings can be harder to match. We want to locate customers with an address on High – and we don't care if it is a street, road, etc. If we try this approach, we also get the Cust 5.

```
Select cust_id, cust_address
From z_wildcards
Where cust_address LIKE '%High%'
;
```

```
   CUST_ID CUST_ADDRESS
---------- ------------------------------
         3 124 High
         4 15 High Road
         5 1554 Rural Highway 12
```

Demo 07: If we try adding a space after the word High, we miss Cust 3.

```
Select cust_id, cust_address
From z_wildcards
Where cust_address LIKE '%High %'
;
```

```
   CUST_ID CUST_ADDRESS
---------- ------------------------------
         4 15 High Road
```

Do not use the LIKE operator if you are doing an exact match. It makes no sense to test for a specific phone number by using the following query. This should be an equality test.

```
Select cust_id, cust_phone
From z_wildcards
Where cust_phone LIKE '510-239-8989';
```

Wildcards are only interpreted if you use the LIKE predicate.

Use the Like operator for string comparisons only. There are safer ways to test numbers and dates. Note there is no problem with testing data that is stored as a string of digits (such as a ssn) with the Like operator.

We will discuss regular expression for testing soon; these provide more precise testing of patterns.

2. Numbers and Like- don't do this!

Use the Like operator for string comparisons only. There are safer ways to test numbers and dates. The following shows some of the issues in testing numbers with wildcards in Oracle.

Demo 08: This will find rows where the list price for a service contains the digit 5. I do not know why anyone would want to do that.

```
Select *
From vt_services
Where srv_list_price like '%5%';
```

Demo 09: This looks like it is trying to find prices that are not even dollar prices- prices that have digits after the decimal point. But because of the way that Oracle casts data types a value such as $45 is not necessarily cast to 45.00. Remember that Like expects to work with strings and numbers are not strings.

```
Select *
From vt_services
Where srv_list_price not like '%.00%';
```

3. Dates and Like- don't do this!

Using Like with date values can also problems. Suppose we want to filter the exam headers tables for certain date components. I will discuss dates and the like operator in another document in this unit.

4. To escape a wildcard:

Since the wildcard is part of the SQL statement and is sent to the server, you cannot escape the wildcard character via SQL*Plus- you have to do this within the SQL statement.

Suppose we had data values that included the % symbol and we wanted to search for those values. The Where clause would start as

```
WHERE   col_1 LIKE  '%%%'
```

But this is an obvious problem since we need to indicate that the middle % is really just a percent symbol and not a wildcard. You need to define an escape character for this query- I am going to use the # character. Now my Where clause looks like this:

```
WHERE   col_1 LIKE  '%#%%' Escape '#'
```

The first % is a wildcard, the second % is the literal character and the third % is again a wild card.

If you are looking for a value which contains two % characters, you could use

```
WHERE   col_1 LIKE  '%#%#%%' Escape '#'
```

We will discuss regular expressions soon which provides for more precise matching.