Table of Contents

1. Dates

Date values are essential to most systems. But date values can be confusing.

1.1. Date values are not stored as strings.

We enter dates using strings and they look like strings when we display them; but dates are dates - not strings.

Oracle has a default format for entering and displaying strings. dd-MON-yy.

Demo 01: set up the following table with a date column and insert dates using various styles.

```
create table   z_tst_dates_0 (
   id integer primary key
 , col_date  date not null
 );
insert into z_tst_dates_0 values(1, '12-JUN-15') ;
insert into z_tst_dates_0 values(2, '12-jun-2015') ;
insert into z_tst_dates_0 values(3, '12-JUN-1915') ;
insert into z_tst_dates_0 values(4, date '1915-06-12') ;
insert into z_tst_dates_0 values(5, date '2015-06-12') ;
```

--This is the default date format for many Oracle systems. Note that rows 3 and 4 appear to be the same value as the other rows.

```
select * from z_tst_dates_0;
```

```
        ID COL_DATE
---------- ---------
         1 12-JUN-15
         2 12-JUN-15
         3 12-JUN-15
         4 12-JUN-15
         5 12-JUN-15
```

-- But if I use a format that includes the full4 digit year, I can see the difference.

```
select id, to_char(col_date, 'yyyy-mm-dd') from z_tst_dates_0;
```

```
        ID TO_CHAR(CO
---------- ----------
         1 2015-06-12
         2 2015-06-12
         3 1915-06-12
         4 1915-06-12
         5 2015-06-12
```

1.1. Date versus DateTime

Oracle does not have a separate date only type. The name of the type is Date- but it always includes both a date component and a time component. The time component is not part of the default display format.

The following expression will include the time component of a date value with a precision of Hour and Minute To_char(ex_date, 'YYYY-MM-DD HH24:mi') We will discuss this function and more formats in another unit.

2. Testing with a Date value

If you are positive that all of the date values for a column were stored with the time component set to midnight then date testing is easier, In the vt_animals table the an_dob all have a time component of midnight. But in the vt_exam_headers tables, the ex_date values have a time component.

Demo 02: These are the rows in the vt_exam_headers table for the month of April 2015.

```
     EX_ID EXAMDATE
---------- -----------------
      2228 2015-04-04  12:30
      2205 2015-04-08  10:30
      2289 2015-04-11  13:00
      2290 2015-04-11  17:00
```

3. Comparing string to string

Demo 03: If I test for exams on April 4, 2015 using date '2015-04-08' or '04-APR-15', I get no matches.

```
select ex_id , ex_date
from vt_exam_headers
where ex_date = date '2015-04-08';
```

```
no rows selected
```

```
select ex_id , ex_date
from vt_exam_headers
where ex_date =  '04-APR-15';
```

```
no rows selected
```

Demo 04: What I need to do is cast the ex_date to a **string** which has the pattern YYYY-MM-DD and then compare that string expression to the proper string literal.

```
select ex_id , ex_date
from vt_exam_headers
where to_char(ex_date, 'YYYY-MM-DD') = '2015-04-08'
;
```

```
     EX_ID EXAMDATE
---------- ---------
      2205 08-APR-15
```

3.1. Using Between with dates

Suppose I want to display all of the exams in the month of Jan 2016.

Demo 05: I could try a Between test but the following will miss the exam on 2016-01-31 9:00 am. If you do not include a time component, then the date value gets a default time component of midnight. The first query shows we do have 8 rows for Jan 2016.

```
select ex_id , ex_date, to_char(ex_date, 'YYYY-MM-DD  HH:Mi')
from vt_exam_headers
order by ex_date desc;
-- selected rows
```

```
     EX_ID EX_DATE   TO_CHAR(EX_DATE,'YYYY-MM-DDHH:MI')
---------- --------- ---------------------------------
      3288 31-JAN-16 2016-01-31  09:00
      3494 22-JAN-16 2016-01-22  09:00
      3325 15-JAN-16 2016-01-15  10:45
      3104 09-JAN-16 2016-01-09  04:30
      4103 08-JAN-16 2016-01-08  03:30
      4102 08-JAN-16 2016-01-08  01:00
      4101 02-JAN-16 2016-01-02  01:00
      3420 01-JAN-16 2016-01-01  04:30
```

```
select ex_id , ex_date
from vt_exam_headers
where ex_date Between date '2016-01-01' and date '2016-01-31';
```

```
     EX_ID EX_DATE
---------- ---------
      4101 02-JAN-16
      4102 08-JAN-16
      4103 08-JAN-16
      3104 09-JAN-16
      3325 15-JAN-16
      3420 01-JAN-16
      3494 22-JAN-16
```

I could try a Between test with the upper range value being '2015-02-01' but if we did have a ex_date of 2015-02-01 midnight, that row would be returned.

Demo 06: A better approach is a compound comparison test; note the comparison operators used.

```
select ex_id , ex_date
from vt_exam_headers
where ex_date >= date '2016-01-01' and ex_date < date '2016-02-01';
```

3.2. Dates and Like

Using Like with date values can also problems. Suppose we want to filter the exam headers tables for certain date components.

Demo 07: We might try the following to find exam dates in Jan 2016.

```
select ex_id , ex_date
From vt_exam_headers
Where ex_date  like '2016-01%';
```

But that does not return any rows- even though we have exams in Jan 2016.

We could use the default Oracle format and let the system do the conversion. This works.

```
select ex_id , ex_date
From vt_exam_headers
Where ex_date  like '%-JAN-16';
```

This also works since we cast the ex_date to a string that matches our wild card pattern.

```
select ex_id , ex_date
```

```
   From vt_exam_headers
   Where to_char(ex_date, 'YYYY-MM-DD')  like '2016-01%';
```

But you might as well do a better string pattern and use an equal tests. Wildcard tests are generally more expensive than equality tests.

```
   select ex_id , ex_date
   From vt_exam_headers
   Where to_char(ex_date, 'YYYY-MM')  = '2016-01';
```

Demo 08: This would find exams done in January of any year. Remember this will be case sensitive

```
   select ex_id , ex_date
   from vt_exam_headers
   where ex_date  like '%JAN%';
```

But it is better to do a more exact pattern and avoid like.

```
   select ex_id , ex_date
   from vt_exam_headers
   where to_char(ex_date, 'MON') = 'JAN';
```

In addition to the problems of matching the default date format for wildcard matching, we do not have a default time format and it is possible that the dba could change the default date format to a different format - such as YYYY-MM-DD- and all code that uses the Like operator with date values will have to be inspected and possibly changed.