Some string- and date- and time-related SQL functions

(adapted from an older version of CS 325 - Reading Packet: "Simple Reports - Parts 1 and 2")

Sources:

- Oracle9i Programming: A Primer, Rajshekhar Sunderraman, Addison Wesley.
- Classic Oracle example tables `empl` and `dept`, adapted somewhat over the years

Below are some Oracle functions related to strings, dates, and times that can be handy in creating more-readable/"prettier" queries and reports. It is not an exhaustive coverage; the goal is to give you some idea of the possibilities (so you can explore further as inspiration strikes you).

NOTE that these can also be called in **PL/SQL statements** as well!

Suggestion:

To get a better feel for these functions and how to use them, I recommend that you have **`sqlplus`** open as you are reading through this, and try the examples using these tables along the way. It is even better if you try out additional calls of these functions as you think of different possibilities for how they might be used.

On nrs-projects.humboldt.edu, you can get a copy of the adapted versions of Oracle tables `empl` and `dept` used in these example by using the following command at the nrs-projects prompt while in the directory you wish to work in:

```
cp ~st10/set-up-ex-tbls.sql .  # notice the SPACE and PERIOD at the end!
```

...and then run this SQL script `set-up-ex-tbls.sql` in `sqlplus`.

Reminder: concatenation

The operator || can be used to combine one or more string literals or columns, projecting the combined result as a single column. So, for example, the following query projects a single column, combining each employee last name, a ', $', and employee salary:

```
select empl_last_name || ', $' || salary "Pay Info"
from empl
order by empl_last_name;
```

Assuming that the `empl` table has the contents inserted by the SQL script `set-up-ex-tbls.sql`, the above query will result in:

```
Pay Info
------------------------------------------------------------
Adams, $1100
Blake, $2850
Ford, $3000
James, $950
Jones, $2975
```

```
King, $5000
Martin, $1250
Michaels, $1600
Miller, $1300
Raimi, $2450
Scott, $3000

Pay Info
----------------------------------------------------------
Smith, $800
Turner, $1500
Ward, $1250

14 rows selected.
```

When creating a report, concatenation can frequently be used to create more-readable results. As just a few examples:

* if you have first and last names for people, and you wish to display them alphabetically (as in a class role, or a phone directory), it looks good to concatenate them last name first, with a comma in-between:

    ```
    select last_name || ', ' || first_name "Name"
    from ...
    where ...
    order by last_name;
    ```

    ...which might look like:

    ```
    Name
    -----------------------------------
    Adams, Annie
    Cartwright, Josh
    Zeff, Pat
    ```

* ...although for a mailing list, or name tags, etc., you'd probably prefer to have the first name first, and maybe you'd even order them by first name:

    ```
    select first_name || ' ' || last_name "Attendees"
    from ...
    where ...
    order by last_name;
    ```

    ...which might look like:

    ```
    Attendees
    -----------------------------------
    Annie Adams
    Josh Cartwright
    Pat Zeff
    ```

* and many combinations of street, city, state, and zip columns are possible:

```
    select city || ', ' ||  state || '  ' || zip
    from ...
    where ...

    select zip || '-' || city
    from ...
    where ...

    select state || ': ' || city
    from ...
    where ...
```

...etc., and these can be ordered by city and then zip, by state and then city and then zip, by zip, by some other column (such as last name or department or salary or hiredate), etc., depending on what is appropriate for that query.

Reminder: date-related function: sysdate

Remember that SQL function `sysdate` returns the current date:

```
insert into empl(empl_num, empl_last_name, job_title, mgr, hiredate, salary,
                 dept_num)
values
('6745', 'Zeff', 'Analyst', '7566', sysdate, 3000, '200');
```

...and the hiredate for Zeff will be the date that this insertion was performed. And `sysdate` can be used in a select as well -- this simply projects the current date for each row in the "dummy" table `dual`, which only has one column and one row, and so simply projects the current date. So if I run the following on February 11, 2025:

```
select sysdate
from dual;
```

....then the result would be:

```
SYSDATE
---------
11-FEB-25
```

Date- and time-related function: `to_char`

Oracle function `to_char` expects a date or a number and a format string, and it returns a character-string version of the given date or number based on that given format.

A complete coverage of all of the possibilities for the format string is beyond the scope of this introduction, but you can easily find out more on the Web. Here are a few examples, though, to give you some ideas of the the possibilities:

For example, this will project just the month of the given date, projecting that month as the entire name of that month:

```
select empl_last_name, to_char(hiredate, 'MONTH') "MONTH HIRED"
from empl;
```

...resulting in:

```
EMPL_LAST_NAME  MONTH HIR
--------------- ---------
King            NOVEMBER
Jones           APRIL
Blake           MAY
Raimi           JUNE
Ford            DECEMBER
Smith           DECEMBER
Michaels        FEBRUARY
Ward            FEBRUARY
Martin          SEPTEMBER
Scott           NOVEMBER
Turner          SEPTEMBER

EMPL_LAST_NAME  MONTH HIR
--------------- ---------
Adams           SEPTEMBER
James           DECEMBER
Miller          JANUARY
Zeff            FEBRUARY

15 rows selected.
```

If you'd like the month with an uppercase first letter and lowercase letter for the rest, use the format string `'Month'` (and here we'll use a column command, too, to get a non-chopped heading):

```
col hiremonth heading "Month Hired" format a11

select empl_last_name "Last Name", to_char(hiredate, 'Month') hiremonth
from empl;
```

...resulting in:

```
Last Name       Month Hired
--------------- -----------
King            November
Jones           April
Blake           May
Raimi           June
Ford            December
Smith           December
Michaels        February
Ward            February
Martin          September
Scott           November
Turner          September

Last Name       Month Hired
--------------- -----------
Adams           September
James           December
Miller          January
Zeff            February
```

```
15 rows selected.
```

These format examples could easily get a bit long-winded, so here are a few more examples all in one query (and some of these also show how you can include some literals in the format strings, too):

```
col mon_year format a8
col long_version format a29
col brief_versn format a17

select to_char(sysdate, 'YYYY') year,
       to_char(sysdate, 'Mon YYYY') mon_year,
       to_char(sysdate, 'MM-DD-YY') num_version,
       to_char(sysdate, 'Day, Month DD, YYYY') long_version,
       to_char(sysdate, 'DY - Mon DD - YY') brief_versn
from   dual;
```

Granted, sometimes you get surprises -- when run on 2025-02-11, the above results in:

```
YEAR MON_YEAR NUM_VERS LONG_VERSION                  BRIEF_VERSN
---- -------- -------- ----------------------------- -----------------
2025 Feb 2025 02-11-25 Tuesday  , February  11, 2025 TUE - Feb 11 - 25
```

I think the "gaps" are based on including the space needed for the "longest" weekday and month names; there are string functions you can use to get rid of such spaces, which we'll discuss shortly, for times when you don't want those gaps.

Here is a summary of some of the available date-related format strings for use in a `to_char` format string:

```
'MM'        - month number
'MON'       - the first 3 letters of the month name, all-uppercase
'Mon'       - the first 3 letters of the month name, mixed case
'MONTH'     - the entire month name, all-uppercase
'Month'     - the entire month name, mixed case
'DAY'       - fully spelled out day of the week, all-uppercase
'Day'       - fully spelled out day of the week, mixed case
'DY'        - 3-letter abbreviation of the day of the week, all-uppercase
'Dy'        - 3-letter abbreviation of the day of the week, mixed case
'DD'        - date of the month, written as a 2-digit number
'YY'        - the last two digits of the year
'YYYY'      - the year written out in four digits
```

even:

```
'D'         - number of date's day in the current week (Sunday is 1)
'DD'        - number of date's day in the current month
'DDD'       - number of date's day in the current year
```

Now, why did I say that `to_char` was a time-related function as well? Because, although it is not obvious, you can store both a date and a time in a column of type `DATE` -- and you can then project the time information of a given date with format strings such as:

```
'HH12'       - hours of the day (1-12)
'HH24'       - hours of the day (0-23)
'MI'         - minutes of the hour
'SS'         - seconds of the minute
'AM'         - displays AM or PM depending on the time
```

...and when I ran the following at about 10:31 am on Tuesday, February 11, 2025:

```
select to_char( sysdate, 'D DD DDD Day, Mon YYYY - HH12 HH24 MI SS AM') "UGLY"
from dual;
```

...the result was:

```
UGLY
--------------------------------------------------------------------------------
3 11 042 Tuesday  , Feb 2025 - 10 10 31 35 AM
```

a few more examples of date-related operations and functions

function `to_date`

Have you noticed yet that the Oracle Date type supports + and -? If you add a number to a date, the result is the date that results from adding that number of days to that date! If run on February 11, 2025, then:

```
select sysdate + 1
from dual;
```

...results in:

```
SYSDATE+1
---------
12-FEB-25
```

Now, you'll find that this addition or subtraction will work fine with a column declared to be a date -- but what if, for whatever reason, you want to add or subtract from a date literal? (Or if you want to use some date function given a date literal?) You'll find that the string that you use for insertion will not work:

```
-- FAILS!!

select '31-DEC-18' + 1
from dual;
```

...with the error message:

```
ERROR at line 1:
ORA-01722: invalid number
```

But:

`to_date` - expects a date-string, and returns the corresponding date

...can allow you to do this: (and this example now demonstrates how, yes, the month and year boundaries are indeed handled reasonably):

```
select to_date('31-DEC-18') + 1
from dual;
```

...results in:

```
TO_DATE('
---------
01-JAN-19
```

function `next_day`

`next_day` - expects a date and a string representing the day of the week, and returns the date of the next date after the given date that is on that day of the week

If you remember that February 11, 2025 was a Tuesday, then:

```
select next_day('11-Feb-2025', 'TUESDAY') nxt_tues,
       next_day('11-Feb-2025', 'MONDAY') nxt_mon,
       next_day('11-Feb-2025', 'FRIDAY') nxt_fri
from dual;
```

...results in:

```
NXT_TUES  NXT_MON   NXT_FRI
--------- --------- ---------
18-FEB-25 17-FEB-25 14-FEB-25
```

functions `add_months` *and* `months_between`

`add_months` - expects a date and a number of months, and results in the date that many months from the given date;

`months_between` - expects two dates, and returns the number of months between those two dates (positive if the first date is later than the second, negative otherwise)

```
select add_months('30-Jan-25', 1) one_mth_later,
       months_between('15-Apr-25', '15-Jan-25') diff1,
       months_between('15-Apr-25', '01-Jun-25') diff2
from dual;
```

...results in:

```
ONE_MTH_L      DIFF1      DIFF2
--------- ---------- ----------
28-FEB-25          3 -1.5483871
```

A few string-related functions

function `initcap`

`initcap` - expects a string, and returns a string with an initial uppercase letter

```
select initcap('SILLY') looky
from dual;
```

...results in:

```
LOOKY
-----
Silly
```

functions `lower` and `upper`

`lower` - expects a string, and returns an all-lowercase version of your string
`upper` - expects a string, and returns an all-uppercase version of your string

```
select lower(empl_last_name), upper(empl_last_name)
from empl
where job_title = 'President';
```

...results in:

```
LOWER(EMPL_LAST UPPER(EMPL_LAST
--------------- ---------------
king            KING
```

functions `lpad` and `rpad`

`lpad` - "left pad" - expects a string, a desired length, and a padding character, and returns a string that is the given string padded on the left with the given padding character to result in a string with the desired length
`rpad` - "right pad" - expects a string, a desired length, and a padding character, and returns a string that is the given string padded on the right with the given padding character to result in a string with the desired length

```
col dots format a12 tru
col huh format a15 tru
col right_justif format a12 tru

select lpad(empl_last_name, 12, '.') dots, rpad(empl_last_name, 15, '?') huh,
       lpad(empl_last_name, 12, ' ') right_justifd
from empl;
```

...results in:

```
DOTS         HUH             RIGHT_JUSTIF
------------ --------------- ------------
........King King???????????         King
```

```
.......Jones Jones??????????        Jones
.......Blake Blake??????????        Blake
.......Raimi Raimi??????????        Raimi
........Ford Ford???????????         Ford
.......Smith Smith??????????        Smith
....Michaels Michaels???????     Michaels
........Ward Ward???????????         Ward
......Martin Martin?????????       Martin
.......Scott Scott??????????        Scott
......Turner Turner?????????       Turner

DOTS         HUH             RIGHT_JUSTIF
------------ --------------- ------------
.......Adams Adams??????????        Adams
.......James James??????????        James
......Miller Miller?????????       Miller
........Zeff Zeff???????????         Zeff

15 rows selected.
```

And, of course, if a function returns a string, then a call to that function can be used wherever a string is permitted, including within another function call:

```
col "Hiredate" format a28

select lpad( to_char(hiredate, 'Day'), 14, ' ') ||
       to_char(hiredate, '- Month YY') "Hiredate"
from empl;
```

...which results in:

```
Hiredate
----------------------------
     Thursday - November  11
     Monday   - April     12
     Wednesday- May       13
     Saturday - June      12
     Monday   - December  12
     Monday   - December  12
     Tuesday  - February  18
     Friday   - February  19
     Friday   - September 18
     Friday   - November  18
     Sunday   - September 19

Hiredate
----------------------------
     Sunday   - September 18
     Sunday   - December  17
     Saturday - January   16
     Thursday - November  19

15 rows selected.
```

functions `ltrim` and `rtrim`

`ltrim` - expects a string, returns that string with any leading blanks (blanks starting the string) removed
`rtrim` - expects a string, returns that string with any trailing banks (blanks ending the string) removed

```
col nicer format a30

select ltrim('   Hi   ') lftchop, rtrim('   Hi   ') rtchop,
       rtrim(to_char(sysdate, 'Day')) || ', ' || rtrim(to_char(sysdate, 'Month'))
          || ' ' || to_char(sysdate, 'DD, YYYY') nicer
from dual;
```

...which, when run on 2025-02-11, resulted in:

```
LFTCH RTCHO NICER
----- ----- ------------------------------
Hi       Hi Tuesday, February 11, 2025
```

functions `length` and `substr`

`length` - expects a string, and returns the number of character in that string (its length)
`substr` - expects a string, the position to start at in that string (where the first character is position 1), and how long a substring is desired, and returns the substring of that length starting at that position.
(if the 3rd argument is omitted, it returns the rest of the string starting at the given position)

```
col abb1 format a3
col rest format a13

select empl_last_name,
       length(empl_last_name) length,
       substr(empl_last_name, 1, 3) abb1,
       substr(empl_last_name, 3) rest
from empl;
```

...which results in:

```
EMPL_LAST_NAME      LENGTH ABB REST
--------------- ---------- --- -------------
King                     4 Kin ng
Jones                    5 Jon nes
Blake                    5 Bla ake
Raimi                    5 Rai imi
Ford                     4 For rd
Smith                    5 Smi ith
Michaels                 8 Mic chaels
Ward                     4 War rd
Martin                   6 Mar rtin
Scott                    5 Sco ott
Turner                   6 Tur rner

EMPL_LAST_NAME      LENGTH ABB REST
--------------- ---------- --- -------------
Adams                    5 Ada ams
```

```
James                         5 Jam mes
Miller                        6 Mil ller
Zeff                          4 Zef ff

15 rows selected.
```

Again, please note: this is not an exhaustive list of the additional functions that Oracle provides. But it hopefully gives you an idea of the rich set of possibilities available.