1. Get the first day of the month

Quickly returns the first day of current month. Instead of current month you want to find first day of month where a date falls, replace SYSDATE with any date column/value.

```
SELECT TRUNC (SYSDATE, 'MONTH') "First day of current month"
    FROM DUAL;
```

2. Get the last day of the month

This query is similar to above but returns last day of current month. One thing worth noting is that it automatically takes care of leap year. So if you have 29 days in Feb, it will return 29/2. Also similar to above query replace SYSDATE with any other date column/value to find last day of that particular month.

```
SELECT TRUNC (LAST_DAY (SYSDATE)) "Last day of current month"
    FROM DUAL;
```

3. Get the first day of the Year

First day of year is always 1-Jan. This query can be use in stored procedure where you quickly want first day of year for some calculation.

```
SELECT TRUNC (SYSDATE, 'YEAR') "Year First Day" FROM DUAL;
```

4. Get the last day of the year

Similar to above query. Instead of first day this query returns last day of current year.

```
SELECT ADD_MONTHS (TRUNC (SYSDATE, 'YEAR'), 12) - 1 "Year Last Day" FROM DUAL
```

5. Get number of days in current month

Now this is useful. This query returns number of days in current month. You can change SYSDATE with any date/value to know number of days in that month.

```
SELECT CAST (TO_CHAR (LAST_DAY (SYSDATE), 'dd') AS INT) number_of_days
  FROM DUAL;
```

6. Get number of days left in current month

Below query calculates number of days left in current month.

```
SELECT SYSDATE,
       LAST_DAY (SYSDATE) "Last",
       LAST_DAY (SYSDATE) - SYSDATE "Days left"
  FROM DUAL;
```

7. Get number of days between two dates

Use this query to get difference between two dates in number of days.

```
SELECT ROUND ( (MONTHS_BETWEEN ('01-Feb-2014', '01-Mar-2012') * 30), 0)
          num_of_days
  FROM DUAL;

OR

SELECT TRUNC(sysdate) - TRUNC(e.hire_date) FROM employees;
```

Use second query if you need to find number of days since some specific date. In this example number of days since any employee is hired.

8. Display each months start and end date upto last month of the year

This clever query displays start date and end date of each month in current year. You might want to use this for certain types of calculations.

```
SELECT ADD_MONTHS (TRUNC (SYSDATE, 'MONTH'), i) start_date,
       TRUNC (LAST_DAY (ADD_MONTHS (SYSDATE, i))) end_date
  FROM XMLTABLE (
          'for $i in 0 to xs:int(D) return $i'
          PASSING XMLELEMENT (
                     d,
                     FLOOR (
                        MONTHS_BETWEEN (
                           ADD_MONTHS (TRUNC (SYSDATE, 'YEAR') - 1, 12),
                           SYSDATE)))
          COLUMNS i INTEGER PATH '.');
```

9. Get number of seconds passed since today (since 00:00 hr)

```
SELECT (SYSDATE - TRUNC (SYSDATE)) * 24 * 60 * 60 num_of_sec_since_morning
  FROM DUAL;
```

10. Get number of seconds left today (till 23:59:59 hr)

```
SELECT (TRUNC (SYSDATE+1) - SYSDATE) * 24 * 60 * 60 num_of_sec_left
  FROM DUAL;
```

Data dictionary queries

11. Check if a table exists in the current database schema

A simple query that can be used to check if a table exists before you create it. This way you can make your create table script rerunnable. Just replace table_name with actual table you want to check. This query will check if table exists for current user (from where the query is executed).

```
SELECT table_name
  FROM user_tables
```

```
 WHERE table_name = 'TABLE_NAME';
```

12. Check if a column exists in a table

Simple query to check if a particular column exists in table. Useful when you tries to add new column in table using ALTER TABLE statement, you might wanna check if column already exists before adding one.

```
SELECT column_name AS FOUND
  FROM user_tab_cols
 WHERE table_name = 'TABLE_NAME' AND column_name = 'COLUMN_NAME';
```

13. Showing the table structure

This query gives you the DDL statement for any table. Notice we have pass ‘TABLE’ as first parameter. This query can be generalized to get DDL statement of any database object. For example to get DDL for a view just replace first argument with ‘VIEW’ and second with your view name and so.

```
SELECT DBMS_METADATA.get_ddl ('TABLE', 'TABLE_NAME', 'USER_NAME') FROM DUAL;
```

14. Getting current schema

Yet another query to get current schema name.

```
SELECT SYS_CONTEXT ('userenv', 'current_schema') FROM DUAL;
```

15. Changing current schema

Yet another query to change the current schema. Useful when your script is expected to run under certain user but is actually executed by other user. It is always safe to set the current user to what your script expects.

```
ALTER SESSION SET CURRENT_SCHEMA = new_schema;
```

Database administration queries

16. Database version information

Returns the Oracle database version.

```
SELECT * FROM v$version;
```

17. Database default information

Some system default information.

```
SELECT username,
       profile,
       default_tablespace,
       temporary_tablespace
  FROM dba_users;
```

18. Database Character Set information

Display the character set information of database.

```
SELECT * FROM nls_database_parameters;
```

19. Get Oracle version

```
SELECT VALUE
  FROM v$system_parameter
 WHERE name = 'compatible';
```

20. Store data case sensitive but to index it case insensitive

Now this ones tricky. Sometime you might querying database on some value independent of case. In your query you might do UPPER(..) = UPPER(..) on both sides to make it case insensitive. Now in such cases, you might want to make your index case insensitive so that they don't occupy more space. Feel free to experiment with this one.

```
CREATE TABLE tab (col1 VARCHAR2 (10));

CREATE INDEX idx1
   ON tab (UPPER (col1));

ANALYZE TABLE a COMPUTE STATISTICS;
```

21. Resizing Tablespace without adding datafile

Yet another DDL query to resize table space.

```
ALTER DATABASE DATAFILE '/work/oradata/STARTST/STAR02D.dbf' resize 2000M;
```

22. Checking autoextend on/off for Tablespaces

Query to check if autoextend is on or off for a given tablespace.

```
SELECT SUBSTR (file_name, 1, 50), AUTOEXTENSIBLE FROM dba_data_files;

(OR)

SELECT tablespace_name, AUTOEXTENSIBLE FROM dba_data_files;
```

23. Adding datafile to a tablespace

Query to add datafile in a tablespace.

```
ALTER TABLESPACE data01 ADD DATAFILE '/work/oradata/STARTST/data01.dbf'
    SIZE 1000M AUTOEXTEND OFF;
```

24. Increasing datafile size

Yet another query to increase the datafile size of a given datafile.

```
ALTER DATABASE DATAFILE '/u01/app/Test_data_01.dbf' RESIZE 2G;
```

25. Find the Actual size of a Database

Gives the actual database size in GB.

```
SELECT SUM (bytes) / 1024 / 1024 / 1024 AS GB FROM dba_data_files;
```

26. Find the size occupied by Data in a Database or Database usage details

Gives the size occupied by data in this database.

```
SELECT SUM (bytes) / 1024 / 1024 / 1024 AS GB FROM dba_segments;
```

27. Find the size of the SCHEMA/USER

Give the size of user in MBs.

```
SELECT SUM (bytes / 1024 / 1024) "size"
  FROM dba_segments
 WHERE owner = '&owner';
```

28. Last SQL fired by the User on Database

This query will display last SQL query fired by each user in this database. Notice how this query display last SQL per each session.

```
SELECT S.USERNAME || '(' || s.sid || ')-' || s.osuser UNAME,
         s.program || '-' || s.terminal || '(' || s.machine || ')' PROG,
         s.sid || '/' || s.serial# sid,
         s.status "Status",
         p.spid,
         sql_text sqltext
    FROM v$sqltext_with_newlines t, V$SESSION s, v$process p
   WHERE     t.address = s.sql_address
         AND p.addr = s.paddr(+)
         AND t.hash_value = s.sql_hash_value
ORDER BY s.sid, t.piece;
```

Performance related queries

29. CPU usage of the USER

Displays CPU usage for each User. Useful to understand database load by user.

```
SELECT ss.username, se.SID, VALUE / 100 cpu_usage_seconds
    FROM v$session ss, v$sesstat se, v$statname sn
```

```
   WHERE      se.STATISTIC# = sn.STATISTIC#
         AND NAME LIKE '%CPU used by this session%'
         AND se.SID = ss.SID
         AND ss.status = 'ACTIVE'
         AND ss.username IS NOT NULL
ORDER BY VALUE DESC;
```

30. Long Query progress in database

Show the progress of long running queries.

```
SELECT a.sid,
         a.serial#,
         b.username,
         opname OPERATION,
         target OBJECT,
         TRUNC (elapsed_seconds, 5) "ET (s)",
         TO_CHAR (start_time, 'HH24:MI:SS') start_time,
         ROUND ( (sofar / totalwork) * 100, 2) "COMPLETE (%)"
    FROM v$session_longops a, v$session b
   WHERE      a.sid = b.sid
         AND b.username NOT IN ('SYS', 'SYSTEM')
         AND totalwork > 0
ORDER BY elapsed_seconds;
```

31. Get current session id, process id, client process id?

This is for those who wants to do some voodoo magic using process ids and session ids.

```
SELECT b.sid,
       b.serial#,
       a.spid processid,
       b.process clientpid
  FROM v$process a, v$session b
 WHERE a.addr = b.paddr AND b.audsid = USERENV ('sessionid');
```

- V$SESSION.SID AND V$SESSION.SERIAL# is database process id
- V$PROCESS.SPID is shadow process id on this database server
- V$SESSION.PROCESS is client PROCESS ID, ON windows it IS : separated THE FIRST # IS THE PROCESS ID ON THE client AND 2nd one IS THE THREAD id.

32. Last SQL Fired from particular Schema or Table:

```
SELECT CREATED, TIMESTAMP, last_ddl_time
  FROM all_objects
 WHERE      OWNER = 'MYSCHEMA'
       AND OBJECT_TYPE = 'TABLE'
       AND OBJECT_NAME = 'EMPLOYEE_TABLE';
```

33. Find Top 10 SQL by reads per execution

```
SELECT *
  FROM (  SELECT ROWNUM,
                 SUBSTR (a.sql_text, 1, 200) sql_text,
                 TRUNC (
                    a.disk_reads / DECODE (a.executions, 0, 1, a.executions))
                    reads_per_execution,
                 a.buffer_gets,
                 a.disk_reads,
                 a.executions,
                 a.sorts,
                 a.address
            FROM v$sqlarea a
        ORDER BY 3 DESC)
WHERE ROWNUM < 10;
```

34. Oracle SQL query over the view that shows actual Oracle connections.

```
SELECT osuser,
         username,
         machine,
         program
    FROM v$session
ORDER BY osuser;
```

35. Oracle SQL query that show the opened connections group by the program that opens the connection.

```
SELECT program application, COUNT (program) Numero_Sesiones
    FROM v$session
GROUP BY program
ORDER BY Numero_Sesiones DESC;
```

36. Oracle SQL query that shows Oracle users connected and the sessions number for user

```
SELECT username Usuario_Oracle, COUNT (username) Numero_Sesiones
    FROM v$session
GROUP BY username
ORDER BY Numero_Sesiones DESC;
```

37. Get number of objects per owner

```
SELECT owner, COUNT (owner) number_of_objects
    FROM dba_objects
GROUP BY owner
ORDER BY number_of_objects DESC;
```

Utility / Math related queries

38. Convert number to words

More info: Converting number into words in Oracle

```
SELECT TO_CHAR (TO_DATE (1526, 'j'), 'jsp') FROM DUAL;
```

Output:

```
one thousand five hundred twenty-six
```

39. Find string in package source code

Below query will search for string ‘FOO_SOMETHING’ in all package source. This query comes handy when you want to find a particular procedure or function call from all the source code.

```
--search a string foo_something in package source code
SELECT *
  FROM dba_source
 WHERE UPPER (text) LIKE '%FOO_SOMETHING%'
AND owner = 'USER_NAME';
```

40. Convert Comma Separated Values into Table

The query can come quite handy when you have comma separated data string that you need to convert into table so that you can use other SQL queries like IN or NOT IN. Here we are converting ‘AA,BB,CC,DD,EE,FF’ string to table containing AA, BB, CC etc. as each row. Once you have this table you can join it with other table to quickly do some useful stuffs.

```
WITH csv
     AS (SELECT 'AA,BB,CC,DD,EE,FF'
                   AS csvdata
           FROM DUAL)
    SELECT REGEXP_SUBSTR (csv.csvdata, '[^,]+', 1, LEVEL) pivot_char
      FROM DUAL, csv
CONNECT BY REGEXP_SUBSTR (csv.csvdata,'[^,]+', 1, LEVEL) IS NOT NULL;
```

41. Find the last record from a table

This ones straight forward. Use this when your table does not have primary key or you cannot be sure if record having max primary key is the latest one.

```
SELECT *
  FROM employees
 WHERE ROWID IN (SELECT MAX (ROWID) FROM employees);

(OR)

SELECT * FROM employees
MINUS
SELECT *
  FROM employees
 WHERE ROWNUM < (SELECT COUNT (*) FROM employees);
```

42. Row Data Multiplication in Oracle

This query use some tricky math functions to multiply values from each row. Read below article for more details.

More info: Row Data Multiplication In Oracle

```
WITH tbl
     AS (SELECT -2 num FROM DUAL
         UNION
         SELECT -3 num FROM DUAL
         UNION
         SELECT -4 num FROM DUAL),
     sign_val
     AS (SELECT CASE MOD (COUNT (*), 2) WHEN 0 THEN 1 ELSE -1 END val
           FROM tbl
          WHERE num < 0)
  SELECT EXP (SUM (LN (ABS (num)))) * val
    FROM tbl, sign_val
GROUP BY val;
```

43. Generating Random Data In Oracle

You might want to generate some random data to quickly insert in table for testing. Below query help you do that. Read this article for more details.

More info: Random Data in Oracle

```
SELECT LEVEL empl_id,
           MOD (ROWNUM, 50000) dept_id,
           TRUNC (DBMS_RANDOM.VALUE (1000, 500000), 2) salary,
           DECODE (ROUND (DBMS_RANDOM.VALUE (1, 2)),  1, 'M',  2, 'F') gender,
           TO_DATE (
                 ROUND (DBMS_RANDOM.VALUE (1, 28))
              || '-'
              || ROUND (DBMS_RANDOM.VALUE (1, 12))
              || '-'
              || ROUND (DBMS_RANDOM.VALUE (1900, 2010)),
              'DD-MM-YYYY')
              dob,
           DBMS_RANDOM.STRING ('x', DBMS_RANDOM.VALUE (20, 50)) address
      FROM DUAL
CONNECT BY LEVEL < 10000;
```

44. Random number generator in Oracle

Plain old random number generator in Oracle. This ones generate a random number between 0 and 100. Change the multiplier to number that you want to set limit for.

```
--generate random number between 0 and 100
SELECT ROUND (DBMS_RANDOM.VALUE () * 100) + 1 AS random_num FROM DUAL;
```

45. Check if table contains any data

This one can be written in multiple ways. You can create count(*) on a table to know number of rows. But this query is more efficient given the fact that we are only interested in knowing if table has any data.

```
SELECT 1
  FROM TABLE_NAME
 WHERE ROWNUM = 1;
```