1. Select a column from a table.

```
select COLUMN --example: first_name
from TABLE;   --example: hr.employees
```

2. Select more than one column from a table.

```
select COLUMN1, COLUMN2
from TABLE;
```

3. Select all columns from a table.

```
select *
from TABLE;
```

4. Operators (+, -, /, *) can be used with columns.

```
select COLUMN*12    -- selected column must be a number (integer || decimal).
from TABLE;         -- operators have precedence:
                            highest ()
                            medium  *  /
                            lowest  +  -
-- if an operation has multiple operators with the same precedence, they are executed left to right.
```

5. Giving an alias to a selected column (nickname).

```
select COLUMN as ALIAS       -- Alias will replace column name in the result.
from TABLE;                  -- place alias in quotation marks if you want to add spaces
                             -- ex. select First_name as “employee’s name”
                             -- keyword ‘as’ is optional.
```

6. Concatenate columns.

```
select COLUMN1 || COLUMN2    -- example: select first_name  || last_name
from TABLE;                  -- there is no space between both names.
```

7. Concatenate columns (add a string in between).

```
select COLUMN1 || 'STRING' || COLUMN2  -- example: select first_name || ' ' || last_name
from TABLE;                            -- added space between both names
```

8. Concatenate columns (add single quotation in between).

```
select COLUMN1 || q '(“STRING”)' ||COLUMN2       -- brackets can be replaced with [], {}
from TABLE;
```

9. Display column’s distinct values.

```
select distinct COLUMN
from TABLE;
```

10. Display table’s description.

```
Describe TABLE;
```

11. Limiting the result (numbers).

```
select COLUMN
from TABLE
where CONDITION-- example: where salary = 5000;
```

12. Limiting results (using logical conditions [‘or’, ‘and’]).

```
select COLUMN
from TABLE
where COLUMN = VALUE
or COLUMN = value;  -- ‘or’ can be replaced with ‘and’
                    -- you can have multiple conditions & use different columns in each one
```

13. Limiting the results (with strings).

```
select COLUMN
from TABLE
where STRING_COLUMN =  'STRING'; -- case sensitive.
```

14. Limiting the results (with dates).

```
select COLUMN
from TABLE
where DATE_COLUMN = 'DATE';   -- format sensitive.
                              -- example where hire_date = '29-jan-05'
```

15. Limiting the results (using comparison operators).

```
select COLUMN
from TABLE
where COLUMN operator CONDITION;       -- example: where salary > 2000
                                       -- operators:
                                                Greater than ‘>’
                                                Greater or equal ‘>=’
                                                Less than ‘<’
                                                Less or equal ‘<=’
                                                Not equal  ‘<>’ ‘!=’ ‘^=’
```

16. Displaying values in a specific range.

```
select COLUMN
from TABLE
where COLUMN between VALUE and VALUE -- example: where salary between 2000 and 3000;
```

17. Displaying values in a specific set.

```
select COLUMN
from TABLE
WHERE COLUMN in (VALUE, VALUE, VALUE, …); -- example: where salary in (2000,2300,10000);
```

18. Display strings that match a pattern.

```
select COLUMN
from TABLE
where STRING_COLUMN like '%STRING PATTERN%';     -- ‘%’ means there can be any number of any letter
                                                 in that position (can be anywhere in the string).
                                                 -- example: where first_name like '%e%'
                                                 -- this means that first name must have an ‘e’.

                                                 -- ‘_‘ means there is a character in this position.
                                                 -- example: where firstname like '_e%'
                                                 -- this means display names where ‘e’ is the second
                                                    Char.
```

19. Display null values.

```
select COLUMN
from TABLE
where COLUMN is NULL; -- example: where commission_pct is NULL
```

20. Display values outside a specific set.

```
Select COLUMN
from HR.EMPLOYEES
where COLUMN not in (VALUE, VALUE, VALUE, …); -- example: where salary not in (2000,2300,10000);
```

precedence.

Operator	Meaning
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical condition
8	AND logical condition
9	OR logical condition

21. Oder results by column.

```
select COLUMN
from TABLE
order by COLUMN;    -- default ordering is ascending (add 'desc' to change it to descending.
                    -- if multiple rows had the same value they will be ordered based on their
                       original Place (problem).
```

22. Order results using columns (solution to prev problem).

```
select COLUMN
from TABLE
order by COLUMN, ALTERNATIVE_COLUMN;    -- if multiple rows had the same value (first order column)
                                           they will be ordered by the alternative column.
```

23. Have the user input sth.

```
select &COLUMN                -- the '&' means a variable and it asks the user to input data/name
from &TABLE                      to that variable.
WHERE &CONDITION              -- to allow the user to enter strings: '&USER_INPUT'
ORDER by &ORDERING_COL;       -- '&' can be placed anywhere (in/as a condition, in/as an order,…).
                              -- Examples:
                                        select &column      user enters: first_name
                                        from &table         user: hr.employees
                                        where &condition    user: employee_id = 30
                                        order by &order     user: last_name
```

24. Reusing a variable.

```
select &&COLUMN
from TABLE
order by &COLUMN;
```

25. Defining a variable.

```
define VARIABLE = COLUMN
select &VARIABLE
from TABLE;
```

26. Change strings to lower case.

```
select lower(COLUMN) -- this function can be used in (select, where, order, inside conditions).
from TABLE;
```

27. Change strings to upper case.

```
select COLUMN
from TABLE
where upper('alex') = COLUMN;   -- example: where upper('K') ||'ing' = last_name;
```

28. Capitalizing the first letter.

```
select COLUMN -- example: select initcap(email)
from TABLE; -- if string has multiple words, it capitalizes the first of each word.
```

29. Concatenate two strings/columns (using built-in function).

```
select concat(COLUMN)        -- example: select concat(first_name, concat(' ',last_name))
from TABLE;                  -- function can only accept two parameters.
```

30. Find substring.

```
select substr(COLUMN,START,END) – START can accept negative values & END is optional.
from TABLE;         -- example: select substr(fist_name,1,1) || substr(last_name,1,1) as initials
                    --  this example displays each employee's initials.
```

31. Find the length of a string.

```
select LENGTH(COLUMN) -- example: where length(first_name) = 4
from TABLE;
```

32. Find the position of a character/substring.

```
select instr(COLUMN, 'CHAR(s)') -- returns the position of the first letter/substring.
from TABLE;                     --  returns 0 it the letter/substring was not found.
```

33. Print in a uniform way (lpad, rpad).

```
select lpad(COLUMN,LENGTH, 'CHAR(s)')  -- fills the empty space to the left with the selected char
from TABLE;                            -- rpad fills the empty space to the right.
                                       -- example: lpad(salary,15, '-')
```

34. Replace a character/substring with a character/substring.

```
select replace(COLUMN, 'CHAR(s)', 'CHAR(s)')     -- example: replace(first_name, 'e', 'X')
from TABLE;                                       -- can work with numbers: replace(salary,0,9)
```

35. Delete leading & trailing characters.

```
select trim('CHAR' from COLUMN)        -- add keyword leading to only delete leading char(s).
from TABLE;                            -- add keyword trailing to only delete trailing char(s).
                                       -- example: trim('e' from first_name)
```

36. Round a number.

```
select round(COLUMN, DECIMAL) -- decimal param is optional. Example: round(105.523,1), round(105.5)
from TABLE;                   -- possible decimal values:
                                   positive (1,2,…): rounds based on the selected decimal (right of
                                                     decimal point.
                                   zero: normal rounding
                                   negative(-1,-2,…): rounds based on the selected integer (left of
                                                      decimal point).
```

37. Truncate a number (round down).

```
select trunc(COLUMN, DECIMAL) -- has the same rules & possible values as the round function.
from TABLE;                    -- rounds the number down based on the given decimal.
```

38. Find the remainder (modulus).

```
select mod(COLUMN, INTEGER) -- example: mod(salary,2) || mod(7,2)
from TABLE;
```

39. Show system date (server side).

```
select SYSDATE
from DUAL;
```

40. Show current date (client side).

```
select CURRENT_DATE -- you can add/sub from current_date || sysdate.
from DUAL;
```

41. Number of months between two dates.

```
select months_between(DATE_COLUMN, DATE_COLUMN)  -- use round to remove all decimal points.
from TABLE;                                      -- example: months_between(sysdate,hire_date)
```

42. Add month(s) to a date.

```
select add_months(DATE_COLUMN,MONTHS) -- doesn't change the data in the database (only the results).
from TABLE;                           -- example: add_months(sysdate,5)
```

43. Display the date of a specific next day.

```
select next_day(DATE_COLUMN, 'DAY')    -- shows the date of the selected day.
from TABLE;                            -- example: next_day(sysdate, 'MONDAY')
```

44. Show the last day of the month.

```
select last_day(DATE_COLUMN)
from TABLE;
```

45.