```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

DATE - format YYYY-MM-DD
DATETIME - format: YYYY-MM-DD HH:MI:SS
SMALLDATETIME - format: YYYY-MM-DD HH:MI:SS
TIMESTAMP - format: a unique number

UPPER() function converts a string to upper-case.

The LOWER() function converts a string to lower-case.

```
-- Convert the current date to YYYY-MM-DD format
SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD') FROM dual;
# 2012-07-19
```

| | |
|---|---|
| 1 | TO_CHAR(SYSDATE, 'YYYY-MM-DD') |
| 2 | TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS') |
| 3 | TO_CHAR(SYSDATE, 'YYYYMMDD') |
| 4 | TO_CHAR(SYSDATE, 'YYYYMM') |
| 5 | TO_CHAR(SYSDATE, 'YYMM') |
| 6 | TO_CHAR(SYSDATE, 'YYYY') |
| 7 | TO_CHAR(SYSDATE, 'YYYY/MM/DD') |
| 8 | TO_CHAR(SYSDATE, 'HH24:MI') |
| 9 | TO_CHAR(SYSDATE, 'HH24:MI:SS') |

```
SELECT COUNT( *) as "Number of Rows"
FROM orders;
```

When the * is used for COUNT(), all records ( rows ) are COUNTed if some content NULL but COUNT(column_name) does not COUNT a record if its field is NULL. See the following examples:

SQL COUNT() function with DISTINCT clause eliminates the repetitive appearance of the same data. The DISTINCT can come only once in a given select statement.

**Syntax :**

```
COUNT(DISTINCT expr,[expr...])
```

The following SQL statement lists the number of customers in each country, sorted high to low:

## Example

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

GROUP BY clause is used with the SELECT statement.
In the query, GROUP BY clause is placed after the WHERE clause.
In the query, GROUP BY clause is placed before ORDER BY clause if used any.

```
SELECT column1, function_name(column2)
FROM table_name
WHERE condition
GROUP BY column1, column2
HAVING condition
ORDER BY column1, column2;

function_name: Name of the function used for example, SUM() , AVG().
table_name: Name of the table.
condition: Condition used.
```

| | |
|---|---|
| 1 | TO_DATE('2012-07-18', 'YYYY-MM-DD') |
| 2 | TO_DATE('2012/07/18', 'YYYY/MM/DD') |
| 3 | TO_DATE('2012-07-18 13:27:18', 'YYYY-MM-DD HH24:MI:SS') |
| 4 | TO_DATE('07/18/2012 13:27:18', 'MM/DD/YYYY HH24:MI:SS') |
| 5 | TO_DATE('17-FEB-2013', 'DD-MON-YYYY') |

The basic syntax of DISTINCT keyword to eliminate the duplicate records is as follows –

```
SELECT DISTINCT column1, column2,.....columnN
FROM table_name
WHERE [condition]
```

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.
The COUNT() function returns the number of rows that matches a specified criteria.

The AVG() function returns the average value of a numeric column.

The SUM() function returns the total sum of a numeric column.

Max is a group function/aggregate function/summary function
Greatest is a single row function.- operates on every row. More than one value/column must be given(otherwise no use!). Mostly used in pl/sql.

```
SELECT CONCAT(Region_Name, Store_Name)
FROM Geography
WHERE Store_Name = 'Boston';
```

```
SELECT Region_Name || ' ' || Store_Name
FROM Geography
WHERE Store_Name = 'Boston';
```

```
NVL (expr1, expr2)
```

**expr1** is the source value or expression that may contain a null.
**expr2** is the target value for converting the null.

```
SELECT  salary, NVL(commission_pct, 0),
    (salary*12) + (salary*12*NVL(commission_pct, 0))
      annual_salary FROM employees;
```

```
NVL2 (expr1, expr2, expr3)
```

**expr1** is the source value or expression that may contain null
**expr2** is the value returned if expr1 is not null
**expr3** is the value returned if expr2 is null

```
SELECT last_name, salary, commission_pct,
 NVL2(commission_pct, 'SAL+COMM', 'SAL')
 income FROM employees;
```

```
DECODE(col|expression, search1, result1
 [, search2, result2,...,][, default])
```

```
-- Specify a datetime string and its exact format
SELECT TO_DATE('2012-06-05', 'YYYY-MM-DD') FROM dual;
```

| Oracle TO_DATE | Format Specifier |
|---|---|
| YYYY | 4-digit year |
| YY | 2-digit year |
| RRRR | 4-digit or 2-digit year, 20th century used for years 00-49, otherwise 19th |
| MON | Abbreviated month (Jan - Dec) |
| MONTH | Month name (January - December) |
| MM | Month (1 - 12) |
| DY | Abbreviated day (Sun - Sat) |
| DD | Day (1 - 31) |
| HH24 | Hour (0 - 23) |
| HH or HH12 | Hour (1 - 12) |
| MI | Minutes (0 - 59) |
| SS | Seconds (0 - 59) |

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

```
SELECT last_name, job_id, salary,
   DECODE(job_id, 'IT_PROG', 1.10*salary,
    'ST_CLERK', 1.15*salary,
    'SA_REP', 1.20*salary,salary)
     REVISED_SALARY FROM employees;
```

| | |
|---|---|
| = | Equal to |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| <> | Not equal to |

| | |
|---|---|
| WHERE CustomerName LIKE 'a%' | Finds any values that starts with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that ends with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a_%_%' | Finds any values that starts with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that starts with "a" and ends with "o" |

## The IS NULL Operator

The IS NULL operator is used to test for empty values (NULL values).

## The IS NOT NULL Operator

The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;
```