SQL - Structured Query Language
For RDBMS.

Main 2 sections:
- DDL: Data definition language
- DML: Data manipulation language
etc.

MySQL - case insensitive language; SELECT, SeleCT, select

BIT   ----- default 1 bit
BIT(30)

BOOL

INT
INT(6)
INT UNSIGNED
INT(6) UNSIGNED

DOUBLE ---- default value
DOUBLE(10,3)

DATE   ----- '2021-07-28'

TIME ----- '23:20:59'

DATETIME ---- '2021-07-28 13:11:30'

string value:
----

characterwise	bytewise
char --- default: 1 char	BINARY --- 1 byte
char(10) --- exact 10 length	BINARY(10) - exact 10 byte
VARCHAR(10) --- maximum 10 length - varying length character	VARBINARY(20)- max 20 byte
LONGTEXT	LONGBLOB

BOOL
INT
DOUBLE
DATE, TIME, DATETIME
CHAR, VARCHAR, LONGTEXT

Industry practice: MySQL Keyword(reserve word) - capital
own declaration - small

```
---------------------------

col1          col2
--------------------
abc           100
def           200
ghi           NULL
jkl          300


name       dob (NOT NULL)
-------------------
abc        NULL     --- error
def        10/01/1990


name       address (DEFAULT 'Dhaka')
-------------------
abc        dhaka
def        comilla
ghi        Dhaka --- will be set by the DBMS
jkl        bbaria
```

MySQL:

Primary Key = (Unique + Not Null)

UNIQUE

```
id     email      name
(pk)   unique
------------------------------
1      abc@gmail.com
2      abc@gmail.com   -- error
```

CHECK(expr)

```
acc_no     balance
           CHECK (balance>500)
-------------------------
1          200
```

Primary Key:

```
id
(PRIMARY KEY)
---------------------------------------
```

Foreign key

id (pk)	name
1	abc
2	def

acc_no	balance	holder_id (fk refer user id column)
a1	1000	1000 x

AUTO_INCREMENT

1 table , 1 auto_increment column exists (not more than that) --- PK

ID (PK) (AUTO_INCREMENT)	name
1	abc
2	def

----

1. Database Schema

create database dbname;
- dbname duplicate error generate.

create database if not exists dbname;
- dbname duplicate no error will be generated.

2. Delete an existing database

drop database dbname;
- if no database found will generate error

drop database if exists dbname;
- if no database found won't generate any error.

3. Database table creation:

MySQL:
------
1. create database (schema):
----
create database dbname;
- duplicate dbname, error show

create database if not exists dbname;
- duplicate dbname ignore

2. delete database:
----
drop database dbname;
- dbname doesn't exist, error show

drop database if exists dbname;
- dbname doesn't exists, no error

----

3. Table creation:
----

user

id	name
1	abc
2	def - x
3	ghi

order
o_id   amount   user_id - foreign key(user_id)
references user(id)
                       ON DELETE SET NULL
                       ON UPDATE RESTRICT|SET
NULL|CASCADE

```
--------------------------
1         1000$      1
2          2000$      NULL
3          3000$      1
4          1500$      NULL
```

table 1 code:

```
CREATE TABLE  IF NOT EXISTS table1(
        colA INT AUTO_INCREMENT,
   colB VARCHAR(40),
   colC DATETIME DEFAULT '2020-01-01 00:00:00',
   foreign_colA int,
   foreign_colP VARCHAR(30), #pending

   CONSTRAINT pk PRIMARY KEY(colA),
   CONSTRAINT fk FOREIGN KEY(foreign_colA)
                               REFERENCES table1(colA)
                               ON DELETE  SET NULL
);
```

```
CREATE TABLE IF NOT EXISTS table2(
        colP VARCHAR(30),
   colQ CHAR(10),
   colR DOUBLE NOT NULL,
   foreign_colA INT,

   CONSTRAINT pk1 PRIMARY KEY(colP),
   CONSTRAINT uq  UNIQUE(colQ),
   CONSTRAINT fk1 FOREIGN KEY(foreign_colA)
                              REFERENCES table1(colA)
                              ON DELETE  CASCADE
);
```

foreign key declare:

```
ALTER TABLE table1
ADD CONSTRAINT fk3 FOREIGN KEY(foreign_colP)
                                      REFERENCES table2(colP)
               ON DELETE SET NULL;




ALTER TABLE table2
ADD COLUMN colS DATETIME NOT NULL DEFAULT
'2021-12-31 00:00:00';

ALTER TABLE table2
DROP COLUMN colS;
```

ALTER TABLE table2
DROP PRIMARY KEY;

ALTER TABLE table2
ADD CONSTRAINT pk2 PRIMARY KEY(colR);

ALTER TABLE table2
DROP INDEX uq;

DROP TABLE IF EXISTS table2;

DML - Data Manipulation Language:
----

1) Data Insert

INSERT INTO tablename VALUES(col1value, col2value, col3value, ... colnvalue);
- all column value insertion

INSERT INTO tablename(col1, col10, col5)
VALUES(col1value, col10value, col5value);
- value insertion to some specific columns

example:

```
INSERT INTO table1 VALUES(NULL, "dbms", "2021-10-01 12:10:20",NULL,NULL),
                    (NULL, "dbms1", "2021-12-01 12:10:20",NULL,NULL)

INSERT INTO table1(colB, foreign_colA, colC)
VALUES("new data",3,'2022-01-01 23:59:59');
```

2) Data Update

```
UPDATE  tablename
SET     col1=col1value, col2=col2value, col10=col10value,
... ...
WHERE   condition;
```

example:

```
UPDATE    table1
SET   colB="new colB value", colC="2025-01-01 12:00:00"
WHERE     colA>2;
```

3) Data Delete

```
DELETE FROM tablename
WHERE   condition;
```

example:

```
DELETE FROM table1
WHERE foreign_colA=1
```

MySQL: equality check = instead of ==

MySQL:

Data Search:
----

1) Full table data read:

```
   SELECT  *
   FROM    tablename;
```

   * = all column
   default = all row

   Example:

```
   SELECT  *
   FROM     job_history;
```

2) Reading specific columns:

```
SELECT  col10, col2, col5, ... coln
FROM    tablename;
```

default = all row

Example:

```
--------
SELECT  employee_id,
        department_id
FROM    job_history;
```

```
table1
------
col1       col2
-----------------
10         100
20         200
30         150
40         160
```

```
SELECT  col1, col2, col1+col2, col1/10
FROM    table1;
```

```
Output(temporary view):
------------------------
col1       col2    col1+col2   col1/10
---------------------------------------
10         100     110         1
20         200     220         2
30         150     180         3
```

```
   40        160    200       4


   /*show all the employees first_name, salary,
commission_pct, total salary from employees table*/

   SELECT  first_name,
           salary,
           commission_pct,
           salary+(salary*commission_pct)

   FROM     employees;
```

3) Column aliasing (renaming):

```
   SELECT  col1 AS new_colname,
           col2,
           col1+col2+col10 AS 'new colname'

   FROM    tablename;
```

   Example:

```
   /*show all the employees first_name, salary,
commission_pct, total salary from employees table*/

   SELECT  first_name,
           salary,
           commission_pct AS 'Commission Percentage',
           salary+(salary*commission_pct) AS Total_Salary

   FROM    employees;
```

4) Showing distinct data (no repeated data):

```
SELECT DISTINCT *|col1, col2, col3+col4 AS 'new colname', ...
FROM          tablename;
```

```
Table1
------
col1      col2
----------------
10        100
20        200
30        150
40        160
10        100
10        120
30        180
40        160
```

```
SELECT  col1, col2
FROM    table1;
```

output:

```
-------
col1      col2
----------------
10        100
20        200
30        150
40        160
10        100
10        120
30        180
40        160
```

```
SELECT  DISTINCT    col1, col2
FROM                table1;
```

output:

```
-------
col1      col2
----------------
10        100
20        200
30        150
40        160
10        120
30        180
```

```
SELECT  col1
FROM    table1;
```

default = all row

```
output:
-------
col1
----
10
20
30
40
10
10
30
40

SELECT DISTINCT  col1
FROM             table1;

default = all row

output:
-------
col1
----
10
20
30
40
```

/*show all the employees department_id from employees table*/

```
SELECT  DISTINCT department_id
```

```
   FROM     employees;
```

5) Showing specific rows:

```
   SELECT [DISTINCT]   *|col1, col2, col3+col4 AS 'new
colname', ...
   FROM             tablename
   WHERE             condition;
```

   Example:

```
   /* Show all the employee details who receives salary
greater than 15000$ */

   SELECT  *
   FROM     employees
   WHERE salary>15000;

   /* Show all the employee details who works in
department number 30 or, 90 or, 100 */

   SELECT  *
   FROM     employees
   WHERE department_id=30 OR department_id=90 OR
department_id=100;

   /* Show all the employee details whose job type is
'IT_PROG' */
```

```
   SELECT  *
   FROM     employees
   WHERE job_id='IT_PROG';
```

   /* Show all the employee details who joined on or after 1990-01-01 in the company */

```
   SELECT  *
   FROM     employees
   WHERE hire_date>='1990-01-01'
```

6) Showing sorted output:

```
   SELECT  [DISTINCT]  *|col1, col2, col3+col4 AS 'new colname', ...
   FROM              tablename
   [WHERE             condition]
   ORDER BY           col1 [ASC|DESC], col3 [ASC|DESC], ...
...
```

   Example:

   /* Show all the employee details based on their salary value from lowest to highest */

```
   SELECT  *
   FROM     employees

   ORDER BY SALARY ASC;
   /* Show all the employees first_name, last_name, email
in the alphabetical order of their first name */

   SELECT  first_name,
           last_name,
           email
   FROM    employees

   ORDER BY first_name ASC

   /* Show all the employees details from senior to junior
*/

   SELECT  *
   FROM    employees

   ORDER BY hire_date ASC;


   table1
   -------
   col1   col2   col3
   --------------------
   abc    10     100
   abc    20     200
   def    30     50
   ghi    40     300
   def    20     150

   jkl    99     999 - x
```

```
SELECT  *
FROM    table1
WHERE   col1!='jkl'
ORDER BY col1 DESC, col2 ASC;

output:
-------
col1   col2   col3
---------------------
ghi    40     300

def    20     150
def    30     50

abc    10     100
abc    20     200
```

7) Limiting number of rows:

- when you need to show some specific rows
- when you can't row filter using WHERE clause

- first 3 seniormost employee details
- the 4th senior most joined on 1990-01-03

```
SELECT  *
FROM    employees
WHERE   hire_date<1990-01-03
```

(wrong approach)

```
Syntax:
-------
SELECT  [DISTINCT]  *|col1, col2, col3+col4 AS 'new colname', ...
FROM              tablename
[WHERE            condition]
[ORDER BY         col1 [ASC|DESC], col3 [ASC|DESC], ... ...]
LIMIT             offset, row_count;


/* Show the top 3 senior most employee details */

SELECT  *
FROM    employees

ORDER BY hire_date ASC
LIMIT   0, 3

/* Show the top 3 lowest salary holder employee details */

SELECT  *
FROM    employees

ORDER BY salary ASC
LIMIT   0, 3
```

```
   /* Show the top 3 lowest salary holder from department
id 90 employee details */

   SELECT  *
   FROM    employees
   WHERE department_id=90
   ORDER BY salary ASC
   LIMIT    0, 3

   /* Show the 5th highest salary holder employee details
*/

   SELECT  *
   FROM    employees

   ORDER BY salary DESC
   LIMIT    4, 1
```

MySQL:

Aggregate Function/Groupwise Function:

SUM(colname)

MIN()
MAX()
AVG()
COUNT()

table1

col1	col2
10	100
20	200
30	300
40	400

SUM(col2)=100+200+300+400=
MIN(col2)=100
MAX(col2)=400
AVG(col2)=(100+200+300+400)/4=
COUNT(col2)=4

output no of rows = no of groups

---

table1

col1	col2	col1+col2
10	100	= 110
20	100	= 120

NULL   400   = NULL
40   400   = 440

```
SELECT  COUNT(*), SUM(col2), MIN(col2), MAX(col2),
AVG(col2), SUM(DISTINCT col2)
FROM    table1;
```

default = all row
default = full table 1 group (you can't call individual columns)

```
Output:
--------

COUNT(col1) SUM(col2)   MIN(col2)   MAX(col2)
AVG(col2)   SUM(DISTINCT col2)
----------------------------------------------------------------------
---------
4         1000       100       400       250          500
```

```
SELECT  SUM(col2), MAX(col1+col2), col1 x
FROM table;
```

```
Output:
--------
SUM(col2)   MAX(col1+col2)   col1(invalid operation)
---------------------------
1000       440          garbage value, 10/20/NULL/40
```

Group formation:

```
SELECT
FROM    tablename
[WHERE  condition]

GROUP BY colname, col1+col2, ... ...

[ORDER BY col1 ASC|DESC, col2 ASC|DESC, ... ...]
[LIMIT  offset, rowcount];
```

```
table1
-------
col1    col2   col1%3
------------
10      100     1  ---- G1
20      200     2  ----------G2
30      300     0  ---------------G3
40      400     1  ---- x ------------------- col1!=40
50      500     2  ----------G2
60      600     0  ---------------G3
```

```
SELECT  COUNT(*), SUM(col2), MAX(col1), col1%3
FROM    table1
WHERE   col1!=40
```

GROUP BY col1%3

(you can't use individual rows after grouping)
(you can't call individual columns after grouping, you must use aggregate functions for different columns)

Output:

```
-------
   COUNT(*)    SUM(col2)   MAX(col1)      col1%3
------------------------------------------------
G1-  1         100        10            1
G2-  2          700        50            2
G3-  2          900        60             0
```

table2

```
-------
col1      col2       col3
----------------------------
abc        10         100     --- G1
abc        20          200     --- G1
def       30         300     ------ G2
ghi       40         400     ---------- G3
def       50         500     ------ G2
abc        10         300     --- G1
def       50         400     ------ G2
ghi       40         1000    ---------- G3
```

SELECT COUNT(*), SUM(col3), col1

```
FROM    table2

GROUP BY    col1;

OUTPUT:
-------
   COUNT(*)       SUM(col3)      col1
----------------------------------------
G1-  3           600           abc
G2-  3            1200          def
G3-  2            1400          ghi


table2
-------
col1       col2       col3
----------------------------
abc        10         100      ---- G1
abc        20         200      ------- G2
def        30         300      ---------- x
ghi        40         400      ---------------G4
def        50         500      --------------------G5
abc        10         300      ---- G1
def        50         400      --------------------G5
ghi        40         1000     ---------------G4

SELECT      COUNT(*), SUM(col3), col1, col2
FROM        table2
WHERE       col1!=def and col2!=30   #single column
condition WHERE
GROUP BY    col1, col2
```

```
HAVING      SUM(col3)>500           #aggregate function
condition HAVING
```

Output:

```
-------
   COUNT(*)    SUM(col3)   col1    col2
---------------------------------------
G1- 2          400        abc     10     - x
G2- 1          200        abc     20     - x

G4- 2          1400       ghi     40
G5- 2          900        def     50
```

condition: COUNT(*)>5 -- having

hire_date>'2021-01-01' -- where

```
/*count department wise employee number*/

SELECT              COUNT(*) AS 'department wise
employee count',
                    department_id

FROM                employees
```

```
GROUP BY        department_id

/*count total number of employees joined per year and
their total salary value */
/*
2021-01-01  -- g1
2021-01-02  -- g2

2021  -- g1
2021  -- g1

*/

SELECT          COUNT(*) AS 'year wise employee
count',
                SUM(salary),
                YEAR(hire_date)

FROM            employees

GROUP BY        YEAR(hire_date)




dept_id        job_id
--------------------------
d1            j1  - g1
d1            j2  ------g2
d1            j1  - g1
```

```
d1              j2  ------g2
d1              j2  ------g2

d2              j3
d2              j4
d2              j4
d2              j3
d2              j3
```

/*
For each department and for each job type, count the total number of employees.
*/

```
SELECT          COUNT(*) AS 'year wise employee
count',
                SUM(salary),
                department_id,
        job_id

FROM            employees

GROUP BY        department_id, job_id;
```

/*
Count region wise total number of countries.
or,
For each region, count number of countries.
*/

```
SELECT          COUNT(*),
                region_id

FROM            countries

GROUP BY        region_id
```

```
hire_date:
----------
(2018-01)-01   --->    g1
2018-02-01     -------------> g2
(2018-01)-12   --->    g1
2018-10-10
2018-02-10     -------------> g2

(2019-01)-01
```

```
YEAR(date), MONTH(date)
```

```
/*
For each year and each month, calculate the average salary
of that year and that month.
*/
```

```
SELECT          AVG(salary),
                YEAR(hire_date), MONTH(hire_date)

FROM            employees

GROUP BY        YEAR(hire_date), MONTH(hire_date)
```

MySQL:

```
SELECT    col1, col2, col3+col4, ...
FROM      tablename
```

JOIN operation:

```
table1 AS t1
------
col1   col2
------------
1      abc
2      def
3      ghi
4      jkl

table2 AS t2
------
col3      col1   fk_col1
--------------------------
100        a      3 - x
```

```
200        b      2 - X
500        c      1 - X
600        d      2 - X
700        e      1 - X
800        f      NULL - x
```

* = all table all column
t1.*=table1 all column
t2.*=table2 all column
t1.col1=table1 col1

Show all the data from table1 and table2.

```
SELECT  *
FROM    table1 AS t1

        JOIN
        table2 AS t2
        ON t2.fk_col1=t1.col1

WHERE   t1.col2!='ghi' and t2.col1!='d'
GROUP BY t1.col1
[HAVING ...]
[ORDER BY ...]
[LIMIT ... ...]
```

Output:

```
-------
t1.col1   t1.col2   -   t2.col3   t2.col1   t2.fk_col1
---------------------------------------------------------
1         abc           500       c      1  ---------G1
```

```
1       abc       700     f     1  ---------G1

2       def       200     b     2  ----------------G2
2       def       600     d     2   - hide

3       ghi       100     a     3   - hide


table1 AS t1
------
col1   col2
------------
1     abc
2     def
3     ghi
4     jkl

table2 AS t2
------
col3      col1   fk_col1
---------------------------
100       a      3 - ok
200       b      2 - ok
500       c      1 - ok
600       d      2 - ok
700       e      1 - ok
800       f      NULL - null(false) x

SELECT  *
FROM    table1 AS t1
```

```
        JOIN
        table2 AS t2
        ON t1.col1>t2.fk_col1
```

```
Output:
-------
t1.col1   t1.col2   -   t2.col3   t2.col1   t2.fk_col1
-----------------------------------------------------------
2         def           500        c       1
2         def           700        e       1

3         ghi           200        b       2
3         ghi           500        c       1
3         ghi           600        d       2
3         ghi           700        e       1

4         jkl           100        a       3
4         jkl           200        b       2
4         jkl           500        c       1
4         jkl           600        d       2
4         jkl           700        e       1
```

```
LEFT JOIN:
----------
table1 AS t1
------
col1   col2
------------
1      abc
2      def
```

```
3      ghi
4      jkl - unused
```

```
table2 AS t2
------
col3       col1    fk_col1
--------------------------
100        a       3
200        b       2
500        c       1
600        d       2
700        e       1
800        f       NULL  - unused
```

```
SELECT  *
FROM    table1 AS t1    #left table

        LEFT JOIN
        table2 AS t2    #right table
        ON t2.fk_col1=t1.col1
```

```
Output:
-------
t1.col1    t1.col2   -    t2.col3    t2.col1    t2.fk_col1
----------------------------------------------------------
1          abc            500        c      1
1          abc            700        f      1

2          def            200        b      2
2          def            600        d      2
```

```
3         ghi          100        a      3

4         jkl          NULL       NULL   NULL
```

Practice:

/* show all the country name and corresponding region name */

```
SELECT    r.region_name,
          c.country_name

FROM      regions AS r

          JOIN
    countries AS c
    ON r.region_id=c.region_id
```

/* show all the city, corresponding country name and corresponding region name */

```
SELECT    r.region_name,
          c.country_name,
    l.city

FROM      regions AS r
```

```
                JOIN
        countries AS c
        ON r.region_id=c.region_id

        JOIN
        locations AS l
        ON      l.country_id=c.country_id
```

/* show all the deparmtent name, corresponding city name, corresponding country name and corresponding region name */

```
SELECT  r.region_name,
        c.country_name,
    l.city,
    d.department_name

FROM    regions AS r

                JOIN
        countries AS c
        ON r.region_id=c.region_id

        JOIN
        locations AS l
        ON      l.country_id=c.country_id

        JOIN
        departments AS d
        ON d.location_id=l.location_id
```

```
/* show the employee id, salary and his colleagues id,
salary who receives higher salary than him

employees AS myself              employees AS colleague
----------
id     salary                id     salary
-------------                 -------------
1      1000                  1      1000 - x
2      2000                  2      2000 - x
3      3000                  3      3000 - ok

output:
-------
myself        colleague
-----------------------
1            2
1            3
2            3

*/

SELECT  myself.employee_id,
        myself.salary,

        colleague.employee_id,
        colleague.salary

FROM    employees AS myself

        JOIN
```

```
employees AS colleague
ON  myself.salary < colleague.salary
```