~~# main string function :-~~

mysql [structured query language]

Tuple :- rows of table

Attributes :- columns of table.

degree :- no. of attributes (columns)

cardinality :- The no. of tuples (rows)

keys :-

* primary key :- A primary key is a set of one or more attributes that can uniquely identify tuples within the relation.

* composite primary key :- when primary key made up of two or more attributes. (column)

* candidate keys :- All attribute combination inside a relation that can serve as primary key. are candidate keys.

* Alternate key :- A candidate key that is not the primary key.

* Foreign key :-
A non-key attribute whose values are derived from the primary key of some other table is called foreign key.

(DDL) [Data definition language]

operations :- create, alter, drop, rename, Truncate etc...

(DML) [Data manipulation language]

operations :- select, update, delete, insert, Lock etc...

Data types :-

* INT (save upto 11 digits), TINYINT (upto 5 digits), SMALLINT, BIGINT (upto 11 digits), FLOAT(m,D), Double(m,D), Decimal(m,D), mediumint (upto 9 digits)

(upto 24 digits)

* DATE :- YYYY-MM-DD [format]

DATETIME :- YYYY-MM-DD HH:MM:SS [format]

* char(m) :-
  └ length

* varchar(m) :-
  └ length

Clauses :- Command consist of one or more logically distinct parts called clauses.

example :-

"FROM salus"

FROM, clauses

Argument

~~Queries~~ :- Simple commands :-

* create database anime;
* show databases;

output :

databases

anime

* use anime; [use <database name> ;]
* create table <table name> (<column name> <data type> [size] <key name>)

example :-

-> create table Dbs (name varchar(20),
Id int (primary key)

* show tables;

Tables_in_anime

Dbs

* Select <column name>, [<column name>,....] From <table name>;

ex:-

```
select name, owner from pet;
```

* Select * from <table_name>; or select All * from <table name>;

<column_n>

ex:-

```
select * from DBS;
```

* Select Distinct <column_name> from <table_name>;

```
select Distinct age from DBS;
```

key point:- only one Null value is returned.

* DESCRIBE | DESC <table_name>;

ex:-

```
DESC DBS;
```

Field	Type	Null	Key	Default	Extra
name	varchar(20)				

* calculation comands:-

select 1+6;

ex:

1+6
7

select 4*3 from dual;

dummy table

outp:-

4*3
12

~~known as~~ provided by mysql.

which contain only one row one column.

* curent date:-

selct curdate();

curdate()
2009-05-03

* scalar expressions:-

select age+10 from DBS;

add operations.

[only for (+), (-), (/), (*), (%)]

add 10 to each value in that column

if Null then, it return, null

* column aliases:-

Select <column_name> AS "Aliases name" from dbs;

ex:- select date, type as "event type" from event;

date	event type
values	values

also,

select 4+3 as "sum" from dual;

op:-

sum
7

* double quotes is used only for more than one word

* Handling nulls:-

select <colum name>, Ifnull (<colum_name>, value

from dbs;

value replace if value is null

ex:-

select name, ~~Id~~ Ifnull (Id, 001) as "Id_Id"

from dbs;

op:-

name	~~Id - Id~~
value	001

* putting text in the query output:-

select name, age, "%" as module from Dbs;

op:-

name	age	module
value	value	% ...

* where - clause :-

Select <column-name>, <column.name>, from <table-name> where <condition>;

ex: select name, aggregate from studen where aggregate > 350;

relational operator

[<=, <, >, >=, =, <> not equal]

logical operator:-

OR (||), AND (&&) ; NOT(!)

ex:-

* select ename, grade from employee where (grade = 'E2' || grade = 'E3');

* select ecode, ename, grade, gross from employee where (grade = 'E4' && gross < 400);

* select ecode, ename from employee where (NOT grade <> 'G1');

* condition based on a range: [Between]

-> select icode, descp from items
where icode between 30 and 50;

(both inclusive)

also,

NOT between

* pattern matches:

-> 'san%' string beginning with 'san'

-> '%idge%' string containing 'idge' ex: 'Ridge'

-> ____ matches string exactly 4 characters

-> ___% matches any string of at least 3 characters.

example:-

-> select firstname, lastname, city from members where pin LIKE '13%';

-> select name from emp where name LIKE '%y';

-> select firstname, city from members where pin NOT like '13%';

-) select * from nit where name LIKE "____";
↳4 (underscore)

* Sorting [order by clause]

select <column name> from <table name>
order by <column name> ASC | DESC

optional
default
ASC

ex:- select name, aggregate from student
where aggregate > 400 order by name;

* Aggregate functions:-

AVG():

SELECT AVG([DISTINCT|ALL], column
-name) From <table>;

COUNT():-

SELECT COUNT({*|[DISTINCT|ALL] column})
From <table>;

max():

select max(DISTINCT|ALL column)
from <table>;

min():-

select min(DISTINCT | ALL column)
from <table>;

sum():-

select sum (DISTINCT | ALL column)
from <table>;

count(<column name>)	count(*)
it not include null	it include null

* Removing data base:

DROP Database <data base_name>;

ex:- Drop database loki;

* Different constraints:-

A constraint is a condition or check applicable on a field or set of fields.

-> unique constraint
-> primary key constraint
-> default constraint.
-> check constraint.
-> foreign key constraint.

* Unique constraint :- [check column contrain only one null & no value repeated]

ex:-

Create table loki (name varchar(20), Id INT NOT NULL UNIQUE);

NOT NULL ↓ No null value

UNIQUE ↓ constraints

* primary key constraint :-

[primary key not allow null value]

ex:-

Create table loki (name varchar(20), Id Int primary key);

also, primary key constraint can set for two columns.

create table loki (f_name varchar(20), l_name varchar(20), primary key (f_name, l_name));

* Default constraint:

if no value is specified then, Default value will be used.

create table loki (name varchar(20), text varchar(20), Default "fail");

* chuck constraint:-

ex:-

```
create table employee (ename varchar(20),
    ecode integer, gross decimal chuck
    (gross > 2000));
```

* foreign key constraint:-

[using primary key as reference from other table to the current table]

ex:-

```
create table items    [parent table]
(  Itemno char(5) NOT null primary key );

create table orders
(  orderno number(6,0) NOT null primary key,
   Itemno char(5) References Items (Itemno));
```

Itemno → parent table

Itemno → primary key of parent table

* (optional)

```
constraint <new_name> <constraint>
```

ex:-

```
create table loki (name varchar(20) constraint
    p_key primary key);
```

* create table from existing table:-

```
create table ardItem AS
(
    Select icode, descp
    from items
    where QOH < ROL
);
```

* SHOW columns from <table_name>;

* inserting values:-

```
insert into loki values('loki', 'fail');
```

also,

```
insert into loki (name, result) values
('loki', 'fail');
```

we can specify the column, if column was missed null will be replaced.

* Insert data from another table:-

```
INSERT INTO <Table-name> select * from
branch2 where gross > 7000.00;
```

ex:-

```
insert into empl(empno, enam, sal)
select enum, Ename, sal
from temp
where months >= 24;
```

* update command :- (update always come with set keyword)

update <table name> set <condition to be replaced> where <to the condition>;

ex:- update items
set ROL = 250;

->* update items
set ROL = 400, QOH = 700
where code < 'I040';

->* update employee
SET gross = gross + 900;

* Deleting Data from table:-

Delete from items;

↓

Delete data but table is exist

ex:-

Delete from employee
where gross < 2200;

* DROP Table:-

Drop table items;

↓ table existence will be erased

* ALTER TABLE:-

-> Adding columns:-

ex:
- ALTER table items Add(P_num int);

-> modifying column definitions:-

* datatypes & size
* Default values
* integrity constraints
* order of column.

ex:- ALTER TABLE EMP1 -> changing
modify (Job char(30)); size & datat

also,

ALTER TABLE Assignment
modify PROJID INT FIRST;

↳ changing calum
Position in 1st

-> changing column name:

ALTER TABLE <TABLENAME>
change <old_column> <new-column>
<datatype-SIZE>;

ex:-

Alter table customers change first_name
firstname varchar(20);

-> Removing table components:-

Alter table customers drop primary key,
drop default, drop column firstname;

* Rename table:-

Rename <old table name> to <new tablename>

* GROUP - BY :-

```
SELECT Job, count(*)
from empl
GROUP BY Job;
```

Job	count(*)
CSE	3
AIDS	2
AIML	4

also,

```
SELECT Job, sum(sal)
from empl.
GROUP BY Job;
```

Job	sum(sal)
CSE	50L
AIDS	40L
AIML	60L

if,

```
SELECT JOB, name
from empl.
GROUP BY Job;
```

Job	name
CSE	LOKI
AIDS	AKI
AIML	MKI

Random names.

* GROUP - HAVING clause.

```
SELECT JOB, count(*)
from empl
GROUP BY JOB
HAVING count(*) < 3;
```

Job	count(*)
~~CSE~~	
AIDS	2

* JOINS :-

Table - A1 → [common column - S_ID] ← Table - A2

S_ID	S_name	age	dob	marks
1	goku	18	2007-03-20	98
2	luffy	19	2006-04-22	97
3	zoro	18	2007-01-22	96
4	Jinwoo	19	2006-03-22	98
5	eren	18	2007-06-12	100
6	mydoria	18	200703-22	97

S_ID	city
1	chennai
2	Kanchipuram
3	Thiruvallur
4	chennai
5	chithabaram
6	erodu

So,

1. select s.s_id, s.s_name, a.city from students as s JOIN stu_location as a where s.s_id = a.s_id;

   we can use "on" also

2. select s.s_id, s.s_name, a.city from students as s INNER JOIN stu_location as a where s.s_id = a.s_id;

3. Let's add one row in students table :- [3 only]

insert into students values(6, 7, 'som', 18, '2007-05-21', 100);

select s.s_id, s.s_name, a.city from students as s left join stu_location as a on s.s_id = a.s_id;

↳ only use "on" not where

4. Let's add one row in stu_location table :-

insert into stu_location values(6, 'Erode');

select s.s_id, s.s_name, a.city from students as s right join stu_location as a on s.s_id = a.s_id;

↳ only use "on" not where.

5. select * from students, stu_location order by students.s_id;

output:

1. [show only related rows & equal rows of two tables]

s_id	s_name	city
1	goku	chennai
2	luffy	kanchipuram
3	zoro	Thiruvallur
4	jinwoo	chennai
5	erin	cithabaram
6	mydoria	erode

2. same output no differences

3.

s_id	s_name	city
1	goku	chennai
2	luffy	kanchipuram
3	zoro	Thiruvallur
4	jinwoo	chennai
5	erin	cithabaram
6	mydoria	erode
7	som	null

[null assigned & show full rows in students table]

5. one row of students is connected with all rows of stu_location table.

So, total no. of rows = $6 \times 6 = 36$

4.

s_id	s_name	city
1	goku	chennai
2	luffy	kanchipuram
3	zoro	Thiruvallur
4	JINWOO	chennai
5	arun	Githabaram
6	mydorial	erode
Null	null	Erokulum

null as assigned & show full rows in stu_location table

MYSQL - PYTHON

* download python with pip module
* & install mysql-python-connection in comand prompt.

pip install mysql-connector-python

so, Then, → connection module.

```
import mysql.connector as m
fo = m.connect("host="Localhost", user=" ", password=" ", database=" ")
```

connection object

optional

m = mysql.connector

& Then,

establish cursor object :-

```
c = fo.cursor()
```

cursor object.

* execute() :-

```
<cursor object>.execute("query")
```

ex:

```
import mysql.connector as m
fo = m.connect(....)
c = fo.cursor()
c.execute("select * from anime")
```

query will execute

* To fetch data from cursor: [<cursor object>.fetch_()]

→ fetchall() = take all rows like :-
[(row1), (row2)...]
↓
list of tuple

-> fetchmany(n)
-> fetchone()

require input.
if fetchmany(3)
[(row1), (row2), (row3)]
↓
list of tuple

take one row as a tuple (row1)

Note:-

if c.fetchone()
↳ take 1st row

So,
c.fetchone()
↳ second time it will 2nd row

↳ same for fetchmany()

Ex:-
```
import mysql.connector as m
fo = m.connect(host="localhost", user="root",
               password="root123", ___)
o = fo.cursor()
o.execute("use lok")
o.execute("select * from students;")
data = o.fetchall()
```
→ [(r1), (r2) ...]

If Db not specified

c. execute ["select name, marks from student;"]
data 2 = c.fetchmany(2) } = 1st two rows
data 3 = c.fetchmany(2) } = 2nd two rows

c. execute ["select age, name from student;"]
data 4 = c.fetchone() } = 1st row
data 5 = c.fetchone() } = 2nd row

use tuple ~~many~~ & list concept to print the information.

fo.close()
↳ close sql..

Note:-

[rows = c.rowcount] total no. of
↳ use to save rows count from cursor (c) to rows]

* For comands like insert, alter, update [manupulation] comands needs commit() before close()

<connection_object>.commit()
<connection_object>.close()

example:-

c.execute("update anime set name='goku' where a_n='obs';")

fo ↳ l.commit()
↳ makes changes in mysql file. If commit() not given · no changes occurs

fo ↳ l.close()

* old-style & new-style format query :-

→ old-style = %(format)

→ new-style = .format( )

example:-(1)

c.execute(" select * from students where name='%s' and marks = %s;"%('LOKI', 99))

note :- [for int no need string quots]

example :- (2)

c.execute(" update students set marks = {} where name = '{}' ; ".format(100, 'LOKI'))

note:- [for int no need string quots]

also,

c.execute("update students set marks = {b} where name = '{a}' ; ".format(a='LOKI', b=99))

a & b assign it self