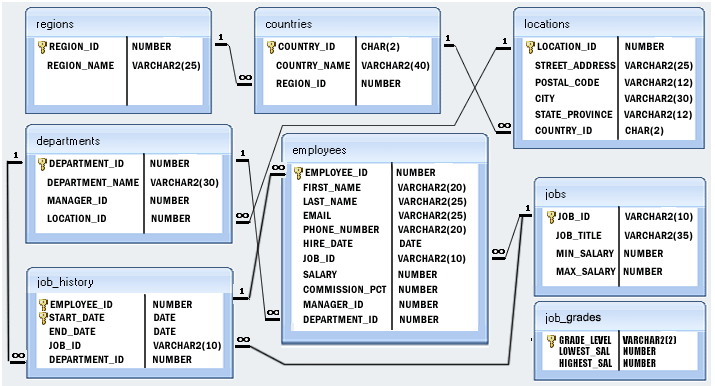
MySQL Notes



1.Create table :

CREATE TABLE IF NOT EXISTS employees (

EMPLOYEE\_ID decimal(6,0) NOT NULL PRIMARY KEY,

FIRST\_NAME varchar(20) DEFAULT NULL,

LAST\_NAME varchar(25) NOT NULL,

JOB\_ID INTEGER NOT NULL,

SALARY decimal(8,2) DEFAULT NULL,

FOREIGN KEY(JOB\_ID)

REFERENCES jobs(JOB\_ID)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

DESC employees;

2. Insert into table

CREATE TABLE IF NOT EXISTS departments (

DEPARTMENT\_ID integer NOT NULL UNIQUE,

DEPARTMENT\_NAME varchar(30) NOT NULL,

MANAGER\_ID integer DEFAULT NULL,

LOCATION\_ID integer DEFAULT NULL,

PRIMARY KEY (DEPARTMENT\_ID)

);

INSERT INTO departments VALUES(60,'SALES',201,89);

INSERT INTO departments VALUES(61,'ACCOUNTS',201,89);

SELECT \* FROM departments;

CREATE TABLE IF NOT EXISTS jobs (

JOB\_ID integer NOT NULL UNIQUE PRIMARY KEY,

JOB\_TITLE varchar(35) NOT NULL DEFAULT ' ',

MIN\_SALARY decimal(6,0) DEFAULT 8000,

MAX\_SALARY decimal(6,0) DEFAULT 20000

);

INSERT INTO jobs(JOB\_ID,JOB\_TITLE) VALUES(1001,'OFFICER');

INSERT INTO jobs(JOB\_ID,JOB\_TITLE) VALUES(1002,'CLERK');

SELECT \* FROM jobs;

CREATE TABLE IF NOT EXISTS employees (

EMPLOYEE\_ID integer NOT NULL PRIMARY KEY,

FIRST\_NAME varchar(20) DEFAULT NULL,

LAST\_NAME varchar(25) NOT NULL,

DEPARTMENT\_ID integer DEFAULT NULL,

FOREIGN KEY(DEPARTMENT\_ID)

REFERENCES departments(DEPARTMENT\_ID),

JOB\_ID integer NOT NULL,

FOREIGN KEY(JOB\_ID)

REFERENCES jobs(JOB\_ID),

SALARY decimal(8,2) DEFAULT NULL

);

INSERT INTO employees VALUES(510,'Alex','Hanes',60,1001,18000);

SELECT \* FROM employees;

INSERT INTO employees VALUES(511,'Tom','Elan',60,1003,22000);

Insert Many at once :

INSERT INTO employees VALUES (100,'Steven','King','SKING','515.123.4567','1987-06-17','AD\_PRES',24000.00,0.00,0,90),

(101,'Neena','Kochhar','NKOCHHAR','515.123.4568','1987-06-18','AD\_VP',17000.00,0.00,100,90),

(102,'Lex','De Haan','LDEHAAN','515.123.4569','1987-06-19','AD\_VP',17000.00,0.00,100,90),

(103,'Alexander','Hunold','AHUNOLD','590.423.4567','1987-06-20','IT\_PROG',9000.00,0.00,102,60),

(104,'Bruce','Ernst','BERNST','590.423.4568','1987-06-21','IT\_PROG',6000.00,0.00,103,60);

3. Update table :

DROP TABLE IF EXISTS region;

CREATE TABLE region(

REGION\_NAME varchar(30)

);

INSERT INTO region VALUES ('HU'),

('JI'),

('MI');

-- SELECT \* FROM region;

UPDATE region SET REGION\_NAME='na';

SELECT \* FROM region ;

-----------------------------------------------------

UPDATE jobs,employees

SET jobs.min\_salary=jobs.min\_salary+2000,

jobs.max\_salary=jobs.max\_salary+2000,

employees.salary=employees.salary+(employees.salary\*.20),

employees.commission\_pct=employees.commission\_pct+.10

WHERE jobs.job\_id='PU\_CLERK'

AND employees.job\_id='PU\_CLERK';

select \* from employees;

4.alter table :

ALTER TABLE countries RENAME country\_new;

-----

ALTER TABLE locations

ADD region\_id INT;

-----

ALTER TABLE locations

ADD ID INT FIRST

------

ALTER TABLE locations

ADD region\_id INT

AFTER state\_province;

-----

ALTER TABLE job\_history

DROP INDEX indx\_job\_id;

-----

ALTER TABLE job\_history

ADD INDEX indx\_job\_id(job\_id);

-----

ALTER TABLE locations DROP PRIMARY KEY;

----

5. query tables , select statements:

SELECT first\_name "First Name", last\_name "Last Name" FROM employees;

----------------------------------

SELECT COUNT(DISTINCT job\_id)

FROM employees;

------------------------------

SELECT MAX(salary), MIN(salary)

FROM employees;

---------------------------

SELECT MAX(salary)

FROM employees

WHERE job\_id = 'IT\_PROG';

-----------------------------

6.Subquery :

a query to find the name (first\_name, last\_name) and the salary of the employees who have a higher salary than the employee whose last\_name='Bull'.

SELECT FIRST\_NAME , LAST\_NAME ,SALARY

FROM employees

WHERE SALARY >

(SELECT SALARY FROM employees WHERE LAST\_NAME = 'Bull');

7. joins

CREATE TABLE IF NOT EXISTS student (

ID integer NOT NULL AUTO\_INCREMENT PRIMARY KEY,

NAME varchar(20) ,

AGE integer ,

CITY integer,

FOREIGN KEY (CITY)

REFERENCES city(cid)

);

-- here i have declared a foreign key which is primary key of another table , here 'city' . I also have a primary key which is different than foreign key .

-- then i declare foreign key with the 'FOREIGN KEY ()' and refer it to the column in the other table with same values or value type . now they these two columns are linked with eachother

CREATE TABLE IF NOT EXISTS city(

CID integer PRIMARY KEY,

CITY varchar(20)

);

INSERT INTO student VALUES(1,"Varun",32,1),(2,"Bhakti",45,2),(3,"Sandeep",67,3),(4,"Kartik",69,2),(5,"Mahesh",98,2),(6,"Suresh",87,5);

INSERT INTO city VALUES(1,"MUMBAI"),(2,"NEW YORK"),(3,"MIAMI"),(4,"DELHI");

SELECT \* FROM student ;

SELECT \* FROM city;

SELECT \*

FROM student

INNER JOIN city

ON student.city = city.cid;

-- returned common records that match in cid and city columns

SELECT \*

FROM student

LEFT JOIN city

ON student.city = city.cid;

-- returns all records from left table (the table left of 'LEFT JOIN') and the matching records from cid and city of the both the tables ie primary and foreign key of the tables

SELECT \*

FROM student

RIGHT JOIN city

ON student.city = city.cid ;

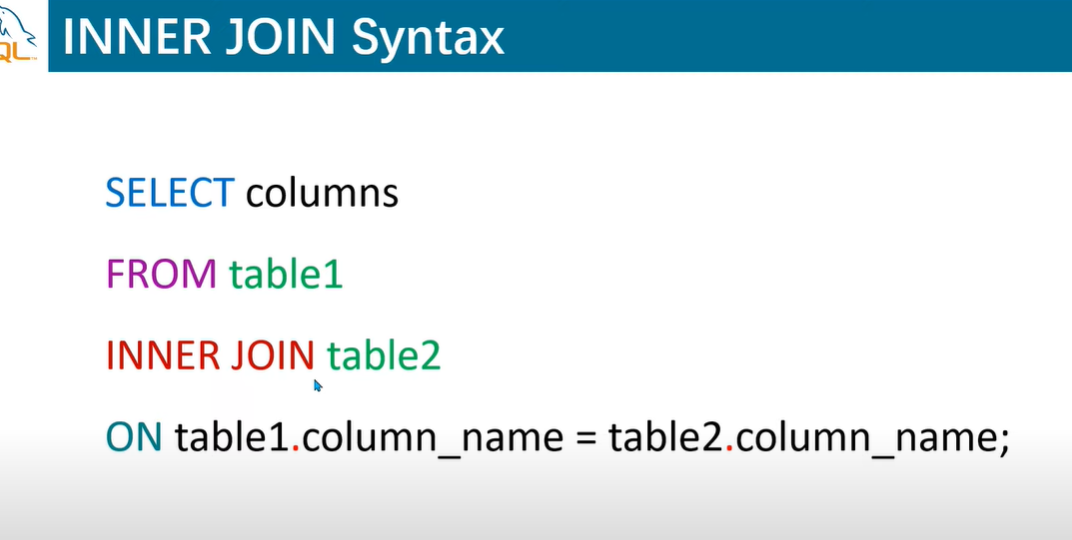
-- returns all the records from the right table (the right of the command RIGHT JOIN) and the ones that match

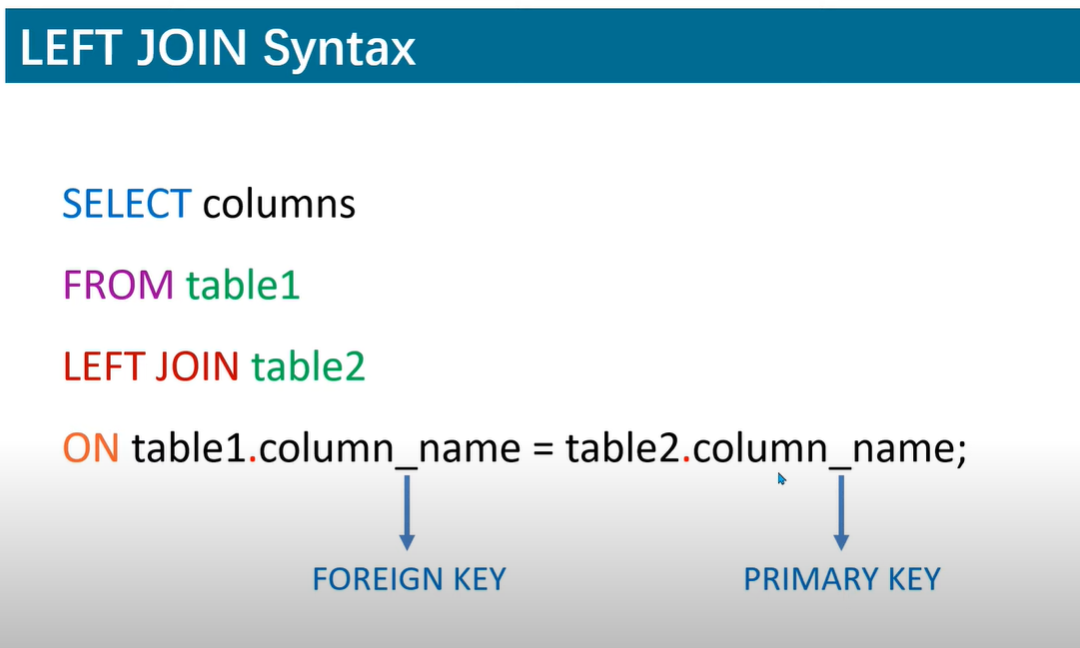
SELECT \*

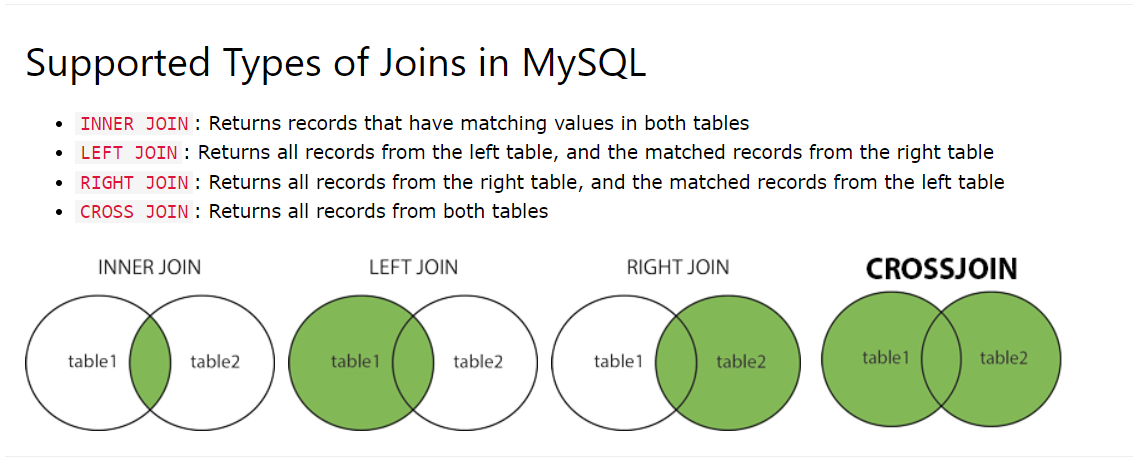
FROM student

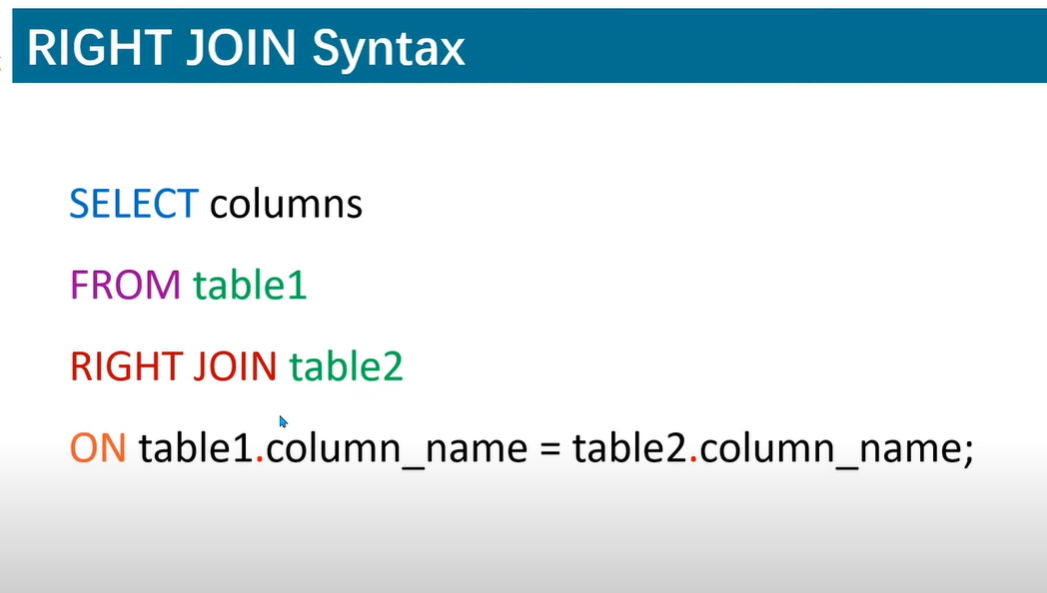
CROSS JOIN city ;

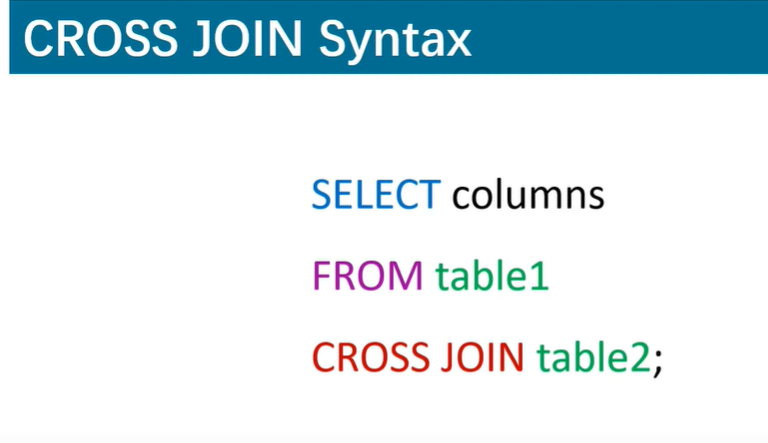
-- returns all



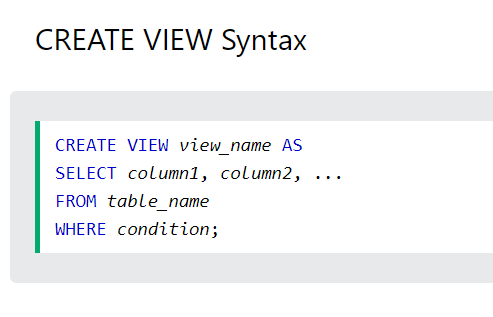








8.View



DROP TABLE IF EXISTS countries;

CREATE TABLE IF NOT EXISTS countries(

ID integer AUTO\_INCREMENT PRIMARY KEY ,

NAME varchar(20)

);

INSERT INTO countries VALUES(1,'USA'),(2,'UK'),(3,'AUS'),(4,'INDIA'),(5,'PAK');

SELECT \* FROM countries;

CREATE VIEW varun AS

SELECT ID , NAME

FROM countries

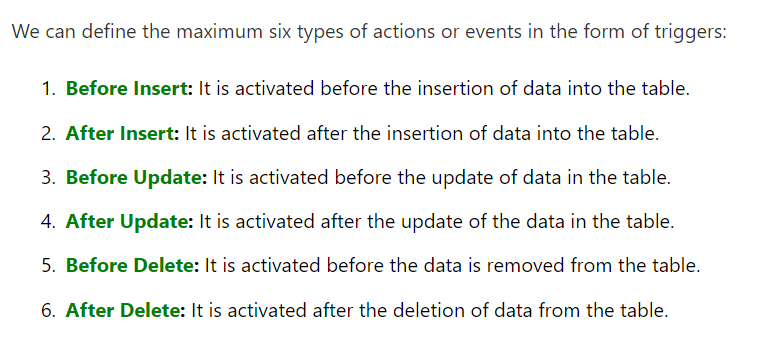
WHERE (ID <3);

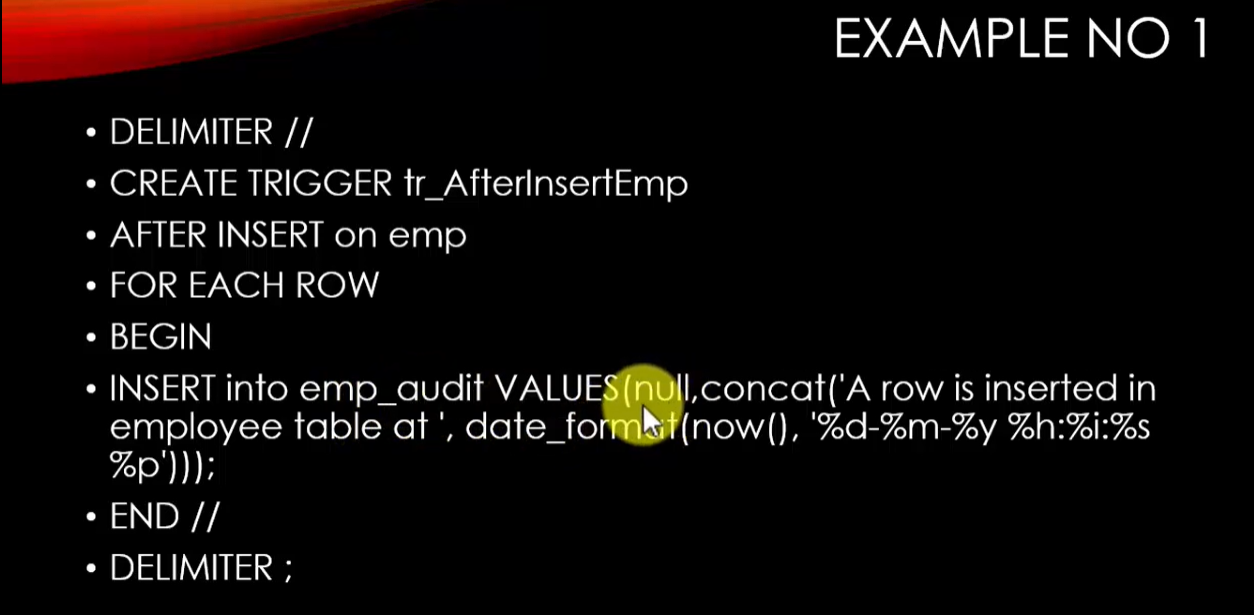
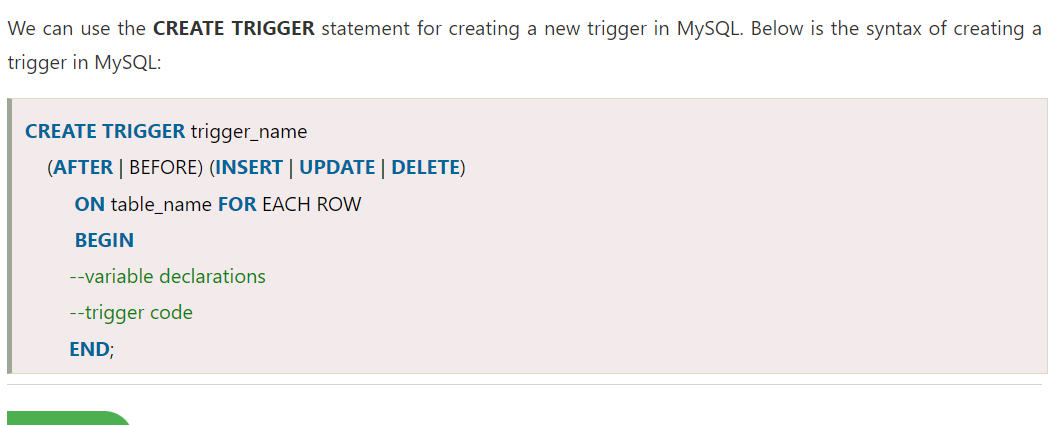
-- view is nothing but 'CREATE VIEW varun AS' . anything ahead of that is just normal querying

SELECT \* FROM varun;

-- this is how you call views

Trigger





|---------------------------------------------------------------------------------------------|

Project Mysql

1.Hotel Management System

We have a main menu where we have options like add room , add customer , modify room , modify customer , room booking , bill generation

First we import mysql.connector library , then we make a mysql.conncetor object called conn, then we make another object called cursor thru which we execute the mysql commands ,

we ask for input such as , for add room , room type , room number , room size etc

then we parse these values into a variable called sql where we use (INSERT INTO table name(column name) )

add customers : use INERT INTO command

modify room : UPDATE table name SET column name = new value

modify customer : UPDATE table name SET column name = new value

room booking :update rooms set status = occupied

insert into booking

bill generation : ask for customer and room id , subtract the date of checking in and current date and multiply with rent – advance

2. Parking Management System

Options : Parking Type , Parking slot , modify type , modify slot , add new vehicle , remove vehicle

3.Weather app using python tkinter and open weather API :

used tkinterlibrary to make the GUI , canvas.geometry(600x800) , padding , font size , font color

get method to data in json format , post to send data , set title ,resizable = false

4.Youtube downloader using tkinter