DATE : 04.09.2024

27.05.2024

About SQL, im repeating for watching the classes again from the day 1.

DAY 1:

1970 by IBM management

Initially name was SEQUL.

Structured English Query Language.

SQL Stands for,

Structured Query Language.

This is used for getting data’s that required

It is that Database Language.

RDBMS – Relational DataBase Managements.

We will manage is commonly in sql is,

Databases,

Tables

And Records.

One database contains en number of tables

Relational databases is, each databases we can connect with an another database with foreign key reference.

The purpose of SQL used to design the managing and quering the data.

DAY-3: 28.05.2024

DDL - Data Definition Language: structuring and managing the data.

Create, alter, drop these are.

DML - Data Manipulation Language: is used to store the data

Change, insert, update, select, store the data, delete,manipulate the data, retrieve the data.

DCL – Data Control Language: used to control the access the database.

Control the access to the database. Permission, revoke permission and all.

Database is collection of data in the structured way

SQL store management studio is a UI which will helps to intract with a mysql server.

COMMENTS: of SQL

Describes the line first then it will not execute by this.

Here,

-- show databases

-- create database fromseppract

-- use fromseppract

-- create table Persons (person\_id int, F\_name varchar(50), L\_name varchar(50), gender varchar(10), qualification varchar(25), mail\_id text)

-- show tables

-- select \* from Persons

-- show databases

-- select \* from Persons

-- show databases

-- show databases

-- create table Persons (person\_id int, F\_name varchar(50), L\_name varchar(50), gender varchar(10), qualification varchar(25), mail\_id text)

-- select \* from Persons

-- show databases

-- show tables

-- create database one

-- use one

-- show tables

-- drop database one

-- use fromseppract

-- show tables

-- select \* from persons

Day 3 - completed

DAY 4: 29.05.24

Truncate - we can remove entire the data what were existing already exept column details.

Drop – we can drop the data fully when you think not use about the data.

-- show databases

-- use fromseppract

-- show tables

-- select \* from persons

/\* here,

im going to create one new from the start.

\*/

-- create table company(first\_name varchar(20), last\_name varchar(20), comp\_age int, comp\_DOB text,

-- comp\_mail text, address text, place varchar(20))

-- show tables

-- select \* from company

/\*

from the existing table, im going to create new table. just like coping the column datas as same.

\*/

Special about we can select any one of column uniquely.

-- create table hotel as select first\_name, last\_name, address, place from company

-- show tables

-- select \* from hotel

-- insert into company(first\_name, last\_name, comp\_age, comp\_DOB,comp\_mail, address, place)

-- values("ve", "che", 25, "12.12.1999", "veche@gmail.com", "abe street", "manamelkudi")

-- select \* from company

-- insert into company(first\_name, last\_name, address, place)

-- values("ohm", "sai", "abcd street", "kaattur")

-- select \* from company

-- select \* from hotel

-- insert into hotel(first\_name, last\_name, address, place)

-- values("Pa", "Ra", "abcde street", "madurai")

-- select \* from hotel

-- insert into hotel values("Ma", "Pa", "efgh", "siruvarai")

-- select \* from hotel

-- select \* from hotel

/\*

here, i want to say something that in hotel table i have inserted two row datas but if you populate now then you could get

output as 4 times which first one row.

why because again and again i have populated the data once i populated already

i dont know that how to be actual result thats come

\*/

-- select \* from hotel

-- select \* from company

-- insert into company values("Che", "Ve", "99", "09.09.1998", "cheve@gmail.com", "xyz street", "nadukudiyiruppu")

-- select \* from company

DAY 3 is completed.

30.05.2024

Today, about to see alter and contrains.

CONSTRAINS:

# PRIMARY KEY

# UNIQUE KEY

# NOT NULL.

And,

# FOREIGN KEY is a reference.

It contais unique and not null.

### NULL – means, there will not be any enter value.

### NOT NULL - means, there will be input we gave.

### Data type TEXT will not allow as primary key or unique key in constrains.

### Primary key will not be duplicates.

### Unique key will give as empty but primary key will not be empty.

### Unique key means every single column should be unique and can have NULL value.

If it is an UNIQUE constrains in the sense you can enter one NULL value and can not allow duplicates value.

If it is an NOT NULL constrains you can not enter even one value also.

PRIMARY KEY is combination of both, you can not enter NOT NULL value and duplicates too.

NOT NULL will allow we to enter as duplicate.

-- today about may 30th class.

-- use fromseppract

-- show tables

-- create table person1(person\_id varchar(20), first\_name varchar(20), last\_name varchar (20), phone\_no text, mail\_id text)

-- select \* from person1

-- describe person1

-- ALTER going to see:

-- alter table person1 add picode int

-- select \* from person1

-- alter table person1 rename column picode to pin\_code

-- select \* from person1

-- alter table person1 add address text

-- select \* from person1

-- alter table person1 drop column address

-- select \* from person1

-- data type conversion:

-- alter table person1 modify phone\_no int

-- describe person1

-- now, going to see about CONSTRAINS:

-- create table candidate(cand\_id int, first\_name varchar(20) NOT NULL, last\_name varchar(20) NOT NULL, phone\_no int NOT NULL,

-- primary key(cand\_id), unique(phone\_no))

-- select \* from candidate

-- describe candidate

-- in this table im going to see, if does allow TEXT type as an PRIMARY KEY?

-- create table candidates(cand\_id int, first\_name varchar(20) NOT NULL, last\_name varchar(20) NOT NULL, phone\_no varchar(50) NOT NULL,

-- primary key(cand\_id), unique(phone\_no))

-- ANSWER is NO.. text type will not accept as an PRIMARY KEY

-- show tables

-- create table candidates1(cand\_id int, first\_name int, last\_name varchar(20) NOT NULL, phone\_no varchar(20) NOT NULL,

-- primary key(first\_name), unique(phone\_no))

-- i tried again to confirm regarding is TEXT type will allow as an primary key - answer is NO.

-- describe candidates1

-- drop table candidates1 ( dropped reason to know regarding to check is required about NOT NULL in primary key)

-- show tables

-- create table candidates1(cand\_id int, first\_name int NOT NULL, last\_name varchar(20) NOT NULL, phone\_no varchar(20) NOT NULL,

-- primary key(first\_name), unique(phone\_no))

-- describe candidates1

-- drop table candidates1

-- create table candidates1(cand\_id int, first\_name int, last\_name varchar(20) NOT NULL, phone\_no varchar(20) NOT NULL,

-- primary key(first\_name), unique(phone\_no))

-- describe candidates1

-- drop table candidates1

-- create table candidates1(cand\_id int, first\_name int, last\_name varchar(20) not null, phone\_no varchar(20) not null,

-- primary key(first\_name), unique(phone\_no))

-- describe candidates1

-- finally, either small or capital letters does not an problem. and primary key will not allow NULL values either you mentioned NOT NULL.

DAY 4 is completed.

31.05.2024:

NOT NULL is you can not use NULL values on specific coloumn.

FOREIGN KEY – will help us to connect one table to another table. (one to many relationship)

(once you connected with foreign key then you can retrieve the data’s of foreign key table)

Primary key contains unique key and NOT NULL.

CONTSTRAINS – is used to specify rules for the data in a table.

PRIMARY KEY – is used to can have for uniquely idendify each records in a table.

## Table with a Foreign key is an Child Table.

## Table with a Primary key is an Parent table or Reference table.

Foreign key – is used to refers the primary key into the another table.

CHECK – is used to limit the value range (control the input that user should insert)

DEFAULT – is used to set the particular value if not exists that value.

AUTO INCREMENT – is used to select the unique number of each records automatically.

(primary key only must be an AUTO INCREMENT)

-- today about may 31st class

-- in this am going to do about connecting both child / parent table. (refference table)

-- show databases

-- use fromseppract;

-- show tables

-- select \* from candidates

-- drop table candidate

-- show tables

-- select \* from candidates

-- drop table candidates

-- select \* from candidates1

-- drop table candidates1

-- select \* from person1

-- drop table persons

-- im going to do the relationship of child and parent table

-- create table persons(per\_id int, per\_name varchar(30), per\_address text, per\_location varchar(20),

-- primary key(per\_id))

-- select \* from persons

-- describe persons

-- create table orders(order\_id int, order\_name varchar(30), order\_address text, orderlocation varchar(30),

-- primary key(order\_id), per\_id int, foreign key(per\_id) references persons(per\_id))

-- select \* from orders

-- describe orders

-- im going to create the same as above for practise

-- create table movies(mov\_id int, mov\_name varchar(30), mov\_hero varchar(30), mov\_heroine varchar(30), primary key(mov\_id))

-- describe movies

-- create table tickets(tic\_id int, tic\_cost int, tic\_class varchar(30),

-- primary key(tic\_id), mov\_id int, foreign key(mov\_id) references movies(mov\_id) )

-- describe tickets

-- DEFUALT:

-- create table fruits1(fru\_id int, fru\_name varchar(30), fru\_colour varchar(30),fru\_price int default 35)

-- insert into fruits1(fru\_id, fru\_name,fru\_colour) values (1,"apple","red")

-- select \* from fruits1

-- create table fruits2(

-- fruits\_name varchar(30),

-- fruits\_colour varchar(30),

-- friuts\_price int

-- default 50 )

-- select \* from fruits2

-- show tables

-- insert into fruits2(fruits\_name, fruits\_colour, friuts\_price) values("orange", "yellow",30)

-- select \* from fruits2

-- insert into fruits2(fruits\_name,fruits\_colour) values ("grapes", "green")

-- insert into fruits2 values ("banana", "yellow", 54)

-- insert into fruits2(fruits\_name,fruits\_colour) values ("lemon", "green")

-- now, im going to give the default value as varchar to confirm wheather is working or not:

-- create table fruits3(

-- fru\_name varchar(30),

-- fru\_colour varchar(30),

-- fru\_price int

-- default "noo"

-- ) IN THIS,DEFUALT WILL ALLOW THE INTEGER TYPE ONLY.

-- practice reason, im going to create the relationship table again.

-- create table college(clg\_id int, clg\_name varchar(30), clg\_location varchar(30), clg\_address text, primary key(clg\_id))

-- create table students(std\_id int, std\_name varchar(30), std\_department varchar(30),std\_year int, clg\_id int, foreign key(clg\_id)

-- references college(clg\_id))

-- select \* from college

-- select \* from students

-- select clg\_id,clg\_name from college

-- show tables

-- describe college

-- describe students

-- CHECK:

-- check is used to give the condition of data. like <,> something.

-- create table personss(pers\_id int, pers\_name varchar(30), pers\_native varchar(30), pers\_age int, check(pers\_age>= 15))

-- select \* from personss

-- insert into personss(pers\_id,pers\_name,pers\_native,pers\_age) value(1,"keegee","france",16)

-- insert into personss values(2,"lofee","india",18)

-- insert into personss values(3,"mofees","italy",13)

-- (this one (value 3) will not be get stored because of i gave lesser value than check constrain)

-- AUTO INCREMENT:

-- (auto increment is used to select the unique numbers of each record automatically)

-- create table personsss(per\_id int not null auto\_increment, per\_name varchar(30),

-- per\_native varchar(30), per\_address text, primary key(per\_id))

-- insert into personsss(per\_name,per\_native,per\_address) values("arun","trichy","abc street"),

-- ("kamalesh","madurai","xsg street"),("balu","salem","hvd street")

-- select \* from personsss

-- create table personssss(per\_id int not null auto\_increment, per\_name varchar(30),per\_age int,

-- primary key(per\_id))

-- insert into personssss(per\_name, per\_age) values("maru",28),("kannan",28),("muthu",24)

-- select \* from personssss

DAY 5 IS COMPLETED.

DAY 6: (03.06.2024)

DATE function starts to insert from year-month-date with double quotes. (“ ”)

In date function, if you given the value between like slash ( / ) but it will take huffing (-) automatically

Mostly month and date we wont use specifically.

Default will works with a table with a column.

-- today about day 6 going to see.

-- show databases

-- use june

-- show tables

-- select \* from friends

-- select \* from highmarks

-- select maths,science from highmarks

-- select \* from movie

-- show databases

-- use fromseppract

-- show tables

-- create table exam\_date(exam\_id int, exam\_name varchar(20), exam\_date date)

-- desc exam\_date

-- insert into exam\_date values(101, "maths", "2024-06-25")

-- insert into exam\_date values(102,"computer","2024/09/24")

-- select \* from exam\_date

-- alter table exam\_date add exam\_new\_date datetime

-- select \* from exam\_date103,"science","2024-06-27","2024-08-24 12:54:02")

-- select \* from exam\_date

-- insert into exam\_date values(

-- insert into exam\_date(exam\_id,exam\_new\_date) values(104,"2024-02-14 09:45:25")

-- select \* from exam\_date

-- AUTO INCREMENT

-- create table tamilnaadu(t\_id int primary key auto\_increment, t\_districts int, t\_mother\_tounge varchar(20))

-- select \* from tamilnaadu

-- insert into tamilnaadu(t\_districts,t\_mother\_tounge) values(38,"tamil")

-- alter table tamilnaadu add t\_capital varchar(20)

-- insert into tamilnaadu(t\_capital) values("chennai")

-- select \* from tamilnaadu

-- in this last table about tamilnaadu, if i add one more column when i inserted all value for existing column

-- then it will appear as second row the new value which i added.

DAY 6 IS COMPLETED.

DAY 7 : (04.06.2024)

SELECT- select we used to select the data from databases.

SELECT- we used to select specific records from the table.

Like,

Column based filter,

Row based filter,

Comparision based filter,

And operator based filter,

Where operator based filter,

Order based filter,

Accending based filter,

Discending based filter.

FILTER THE RECORDS BASED ON COLUMN:

Distinct – is we asking to show unique values by the particular column we select.

Count(distinct) - will helps us to find about how many number of values is there in that particular column.

FILTER THE RECORD BASED ON ROW:

Where – extract only where the condition fulfilled.

ORDER BY – is used to sort the record we expecting. (wise accending order)

-- use fromseppract

-- show tables

-- select \* from animals

-- show databases

-- use june

-- show tables

-- select \* from batch\_std

-- select distinct english from batch\_std

-- select count(distinct maths) from batch\_std

-- select count(name) from batch\_std

-- select count(maths) from batch\_std

-- select \* from batch\_std

-- WHERE:

-- select \* from batch\_std where name = "deepak"

-- select \* from batch\_std where not id=4

-- select \* from batch\_std where maths=75

-- select \* from batch\_std

-- select name,science from batch\_std where maths=44

-- select id,name,maths from batch\_std where not science=78

-- select id,name from batch\_std

-- OR:

-- select \* from batch\_std where name="karthi" or name="parthi"

-- select maths,science from batch\_std where name="raj" or name="gokul"

-- AND:

-- select \* from batch\_std where id=1 and name="deepak"

-- select \* from batch\_std where name="rajan" and science=72

-- select \* from batch\_std where name="gokul" and maths=49

-- show databases

-- show tables

-- select \* from candidates

-- select \* from friends

-- select \* from friends where frnd\_Fname="hari" and frnd\_mail\_id="hari@gmail.com"

-- select \* from friends where frnd\_Fname="cristo" and frnd\_mail\_id="sanju@gmail.com"

-- in line, 43 about will give an output as empty.

-- select \* from friends where frnd\_Fname <> 4

-- select \* from friends where frnd\_salary >=25000

-- select \* from friends where frnd\_salary <= 25000

-- select frnd\_id,frnd\_mail\_id,frnd\_hometown from friends where frnd\_id>=4

-- ORDER BY:

-- select \* from friends order by frnd\_Fname

-- select \* from friends order by frnd\_Fname desc

-- select frnd\_salary,frnd\_address from friends order by frnd\_age

-- select \* from friends

-- select \* from friends order by frnd\_hometown

-- select \* from friends order by frnd\_mail\_id

-- select \* from friends order by frnd\_mail\_id desc

DAY 7 is completed.

DAY 8: (05.06.2024)

-- use fromseppract

-- show tables

-- use june

-- show tables

-- select \* from batch\_std

-- select \* from batch\_std where not science = 78

-- select id,maths,english from batch\_std where name="vivek"

-- select \* from friends

-- select \* from friends order by frnd\_hometown

-- select \* from friends order by frnd\_hometown desc

-- DAY 8: (05.06.2024)

-- use fromseppract

-- show tables

-- select \* from movies

-- drop table movies

-- describe movies

-- select \* from tickets

-- describe tickets

-- drop table tickets

-- drop table movies

-- create table movies(mov\_id int, mov\_name varchar(30), release\_date datetime, collection int)

-- alter table movies add column mov\_hero varchar(30)

-- alter table movies add column heroine\_name varchar(30)

-- select \* from movies

-- alter table movies rename column mov\_hero to hero\_name

-- select \* from movies

-- insert into movies values(1,"marudhu","2018-09-09","20000","vishal","sridivya")

-- select \* from movies

-- insert into movies values(2,"marudhupandi","1995-09-19","1000","ramki","seetha"),(3,"leo","2023-05-26","100000","vijay","trisha"),

-- (4,"dhunivu","2022-03-19","100000","ajith","manju warrier"),(5,"jailer","2024-02-28","200000","rajini","ramyakrishnan")

-- select \* from movies

-- drop table movies

-- (due to executed twise the value so dropped and going to genarate again)

-- create table movies(mov\_id int, mov\_name varchar(30), release\_date datetime, collection int)

-- alter table movies add column mov\_hero varchar(30)

-- alter table movies add column heroine\_name varchar(30)

-- insert into movies values(1,"marudhu","2018-09-09","20000","vishal","sridivya")

-- insert into movies values(2,"marudhupandi","1995-09-19","1000","ramki","seetha"),(3,"leo","2023-05-26","100000","vijay","trisha"),

-- (4,"dhunivu","2022-03-19","100000","ajith","manju warrier"),(5,"jailer","2024-02-28","200000","rajini","ramyakrishnan")

-- select \* from movies

-- insert into movies values(6,"raayan","2024-07-19","50000","dhanush","")

-- select \* from movies

-- insert into movies values(7,"lubber panthu","2024-10-06","50000","gethu\_dinesh","swasika"),

-- (8,"ratchasan","2019-04-03","30000","vishnu\_vishal","amala pual"),(9,"psycho","2020-07-07","35000","udhayanidhi","aditi rao"),

-- (10,"doctor","2022-09-19","500000","sivakarthikeyan","priyanga mohan")

-- select \* from movies

-- select \* from movies where collection = 1000

-- select \* from movies where collection < 100000

-- select \* from movies where mov\_name = "doctor"

-- select distinct mov\_name from movies

-- select count(distinct heroine\_name) from movies

-- select count(heroine\_name) from movies

-- select count(mov\_hero) from movies

-- alter table movies rename column mov\_hero to hero\_name

-- select \* from movies

-- select \* from movies order by heroine\_name desc, hero\_name asc (it wont take the second velue for result.)

-- select \* from movies order by hero\_name desc, heroine\_name asc (it wont take the second velue for result.)

-- select \* from movies order by collection asc

-- select \* from movies order by collection asc, mov\_name desc (it wont take the second velue for result.)

-- select \* from movies where mov\_name like ("%r")

-- select \* from movies where mov\_name like ("m%")

-- select \* from movies where mov\_name like ("%o%")

-- select \* from movies where heroine\_name like ("%d%")

-- select \* from movies where heroine\_name like ("a%")

-- select \* from movies

-- select \* from movies where release\_date = "2022-10-30" or collection = "50000" or hero\_name like ("%m")

-- select \* from movies where hero\_name ="rajini" and heroine\_name = "ramyakrishnan"

-- select mov\_id from movies where hero\_name = "rajini" and heroine\_name = "ramyakrishnan"

DAY 9: (07.06.2024)

AND gate, OR gate are all needed that is compulsory they need one and more value to compare always.

NOT gate where need one value. Does not need more than one value.

-- use fromseppract

-- show tables

-- use june

-- show tables

-- select \* from batch\_std

-- select \* from batch\_std where not science = 78

-- select id,maths,english from batch\_std where name="vivek"

-- select \* from friends

-- select \* from friends order by frnd\_hometown

-- select \* from friends order by frnd\_hometown desc

-- use fromseppract

-- show tables

-- select \* from movies

-- drop table movies

-- describe movies

-- select \* from tickets

-- describe tickets

-- drop table tickets

-- drop table movies

-- create table movies(mov\_id int, mov\_name varchar(30), release\_date datetime, collection int)

-- alter table movies add column mov\_hero varchar(30)

-- alter table movies add column heroine\_name varchar(30)

-- select \* from movies

-- alter table movies rename column mov\_hero to hero\_name

-- select \* from movies

-- insert into movies values(1,"marudhu","2018-09-09","20000","vishal","sridivya")

-- select \* from movies

-- insert into movies values(2,"marudhupandi","1995-09-19","1000","ramki","seetha"),(3,"leo","2023-05-26","100000","vijay","trisha"),

-- (4,"dhunivu","2022-03-19","100000","ajith","manju warrier"),(5,"jailer","2024-02-28","200000","rajini","ramyakrishnan")

-- select \* from movies

-- drop table movies

-- (due to executed twise the value so dropped and going to genarate again)

-- create table movies(mov\_id int, mov\_name varchar(30), release\_date datetime, collection int)

-- alter table movies add column mov\_hero varchar(30)

-- alter table movies add column heroine\_name varchar(30)

-- insert into movies values(1,"marudhu","2018-09-09","20000","vishal","sridivya")

-- insert into movies values(2,"marudhupandi","1995-09-19","1000","ramki","seetha"),(3,"leo","2023-05-26","100000","vijay","trisha"),

-- (4,"dhunivu","2022-03-19","100000","ajith","manju warrier"),(5,"jailer","2024-02-28","200000","rajini","ramyakrishnan")

-- select \* from movies

-- insert into movies values(6,"raayan","2024-07-19","50000","dhanush","")

-- select \* from movies

-- insert into movies values(7,"lubber panthu","2024-10-06","50000","gethu\_dinesh","swasika"),

-- (8,"ratchasan","2019-04-03","30000","vishnu\_vishal","amala pual"),(9,"psycho","2020-07-07","35000","udhayanidhi","aditi rao"),

-- (10,"doctor","2022-09-19","500000","sivakarthikeyan","priyanga mohan")

-- select \* from movies

-- select \* from movies where collection = 1000

-- select \* from movies where collection < 100000

-- select \* from movies where mov\_name = "doctor"

-- select distinct mov\_name from movies

-- select count(distinct heroine\_name) from movies

-- select count(heroine\_name) from movies

-- select count(mov\_hero) from movies

-- alter table movies rename column mov\_hero to hero\_name

-- select \* from movies

-- select \* from movies order by heroine\_name desc, hero\_name asc (it wont take the second velue for result.)

-- select \* from movies order by hero\_name desc, heroine\_name asc (it wont take the second velue for result.)

-- select \* from movies order by collection asc

-- select \* from movies order by collection asc, mov\_name desc (it wont take the second velue for result.)

-- select \* from movies where mov\_name like ("%r")

-- select \* from movies where mov\_name like ("m%")

-- select \* from movies where mov\_name like ("%o%")

-- select \* from movies where heroine\_name like ("%d%")

-- select \* from movies where heroine\_name like ("a%")

-- select \* from movies

-- select \* from movies where release\_date = "2022-10-30" or collection = "50000" or hero\_name like ("%m")

-- select \* from movies where hero\_name ="rajini" and heroine\_name = "ramyakrishnan"

-- select mov\_id from movies where hero\_name = "rajini" and heroine\_name = "ramyakrishnan"

-- (class practise next day)

select \* from movies

-- insert into movies values(11,"money heist","2021-09-29","200000","maru","kallachi")

-- select \* from movies

-- (due to button press mistake some executed values are erased but exist.)

-- select \* from movies

-- select distinct hero\_name from movies

-- select count(hero\_name) from movies

-- select \* from movies

-- select \* from movies where collection > "200000"

-- select \* from movies

-- select release\_date,collection from movies where mov\_name = "psycho"

-- select distinct mov\_name from movies

-- select hero\_name, heroine\_name from movies where mov\_name = "lubber panthu"

-- select mov\_name, collection from movies where mov\_id = "5" or mov\_id = "10"

-- select mov\_name, hero\_name from movies where collection = "500000"

-- select \* from movies

-- show databases

-- use june

-- show tables

-- select \* from senoir\_sal

-- show tables

-- select \* from peoples

-- select \* from peoples order by gender = "female"

-- select \* from peoples

-- select \* from peoples where gender = "female"

-- select \* from peoples

-- select p\_Fname,gender,age from peoples where p\_motherTounge = "tamil"

-- select \* from peoples

-- select \* from peoples where gender = "female" and age >=21

-- select \* from peoples where age >= 21 and gender = "female"

-- select \* from peoples

-- select age,p\_Fname from peoples where gender = "female" and p\_maritalStatus = "single"

-- select p\_Fname,p\_qualif from peoples where gender = "male"

Talk about one more operator which we going to see,

MEMBERSHIP OPERATORS:

Commonly we manage IN alone.

Membership does regarding to know if values exists within specified set values.

-- use fromseppract

-- show tables

-- select \* from movies

-- select mov\_name, collection from movies where heroine\_name = "trisha"

-- select \* from movies where not hero\_name = "vishal"

-- select \* from movies where not hero\_name like "%n%"

-- select \* from movies

-- select \* from movies where collection between 30000 and 100000

-- select \* from movies where not collection >50000

-- select \* from movies

-- use june

-- show tables

-- select \* from peoples

-- select \* from batch\_std

-- select \* from friends

-- select \* from friends where phn\_no is null

-- select \* from friends where phn\_no is not null

-- select \* from friends

-- select \* from friends where frnd\_salary between 25000 and 40000

-- select \* from friends where not frnd\_job = "java developer"

-- select \* from friends

-- MEMBERSHIP OPERATORS:

-- select \* from friends

-- select \* from peoples

-- show databases

-- use may28

-- show tables

-- show databases

-- use junemmonth

-- show tables

-- show databases

-- use may\_month

-- show tables

-- show databases

-- use fromseppract

-- show tables

-- select \* from company

-- select \* from animals

-- select \* from college

-- show tables

-- select \* from movies

-- select \* from movies where hero\_name in ("vishal","vijay","simbu","surya","sivakarthikeyan")

-- show databases

-- use may29

-- show tables

-- select \* from maru

-- select \* from pandi

-- select \* from student

-- select \* from person

-- select \* from maru

-- drop table pandi

-- drop table student

-- select \* from maru

-- insert into maru values(1,"rio","raj",25,"rio@gmail.com","male","abc street")

-- select \* from maru

-- insert into maru values(2,"milan","keen",29,"milan@gmail.com","male","hjbfd street"),

-- (3,"mandee","madh",36,"maddy@gmail.com","male","nygbd street"),(4,"swansika","peth",30,"swansee@gmail.com","female","magfbyew street"),

-- (5,"dulbeen","karow",27,"dulbeen@gmail.com","male","suvusee street"),(6,"bavi","dhaal",22,"baveedhal@gmail.com","female","beestee street")

-- select \* from maru

-- select \* from maru where age in("20","25","23","29","36")

DAY 9 IS COMPLETED.

DAY 10: (10.06.2024)

UPDATE – In this, we can replace the particular value instead of exists value if need.

-- show databases

-- use may\_month

-- show tables

-- use may

-- show tables

-- use junemmonth

-- show tables

-- show databases

-- use june

-- show tables

-- select \* from peoples

-- set sql\_safe\_updates=0

-- use june

-- select \* from peoples

-- update peoples set p\_Fname ="raani",p\_Lname ="shah" where id=1

-- select \* from peoples

-- create table user(id int, name varchar(60), address text,gender varchar(30))

-- select \* from user

-- insert into user values(1,"user1","abc street","nil"),(2,"user2","abc street","nil"),

-- (3,"user3","abc street","nil"),(4,"user4","abc street","nil"),(5,"user1","abc street","nil"),(6,"user1","abc street","nil"),

-- (7,"user1","abc street","nil"),(8,"user1","abc street","nil"),(9,"user1","abc street","nil")

-- select \* from user

-- update user set name="keemon",address="abe street", gender = "male"

-- select \* from user

-- update user set name="punitha",address="glaveedy street", gender = "female" where id=1

-- select \* from user

-- update user set name="kalveem",address="milaadi street", gender = "male" where id = 5 or id = 7

-- select \* from user

-- update user set name = "pallavi", gender = "female" where id=9

-- select \* from user

-- select \* from user

-- show databases

-- select \* from peoples

-- set sql\_safe\_updates = 0

-- update peoples set email\_id = "raanishah@gmail.com", p\_maritalStatus= "married" where age = 19

-- select \* from peoples

-- select \* from peoples where age in ("18","25","19","32")

-- show tables

-- select \* from candidates

-- alter table candidates add column age int

-- alter table candidates add column hometown varchar(30)

-- select \* from candidates

-- describe candidates

-- alter table candidates drop column last\_name

-- select \* from candidates

-- alter table candidates rename column first\_name to cand\_name

-- select \* from candidates

-- select \* from candidates

-- alter table candidates modify column phn\_no varchar(12)

-- select \* from candidates

-- describe candidates

-- insert into candidates values(1,"rahul","6954781236",25,"delhi")

-- select\* from candidates

DAY 10 is completed.

DAY 11: (11.06.2024)

-- use june

-- select \* from peoples

-- create table officials(id int, name varchar(30), department varchar(30), job varchar(30), salary int)

-- insert into officials values(1,"ajay","front office","organiser", 25000),(2,"balan","car delivery","co-ordinator", 30000),

-- (3,"chandru","accounts","accountant", 27000),(4,"daamu","accessories","acc incharge", 22000),(5,"elango","spares","parts assistant", 18000),

-- (6,"faruk","human resources","HR manager", 40000),(7,"ganapathy","house keeping","supervisor", 30000),(8,"hameen","pantry","helper", 18000),

-- (9,"iraivaa","sales","CEO", 300000)

-- select \* from officials

-- select distinct job from officials

-- select count(distinct job) from officials

-- select \* from officials

-- select distinct name from officials

-- select count(distinct name) from officials

-- insert into officials values(10,"jacky","","", 25000)

-- select \* from officials

-- insert into officials values(11,"karthik","car movement","driver", 27000),(12,"lokesh","back office","desk supporter", 26000),

-- (13,"marudhu","bussiness view","data scientist", 900000),(14,"naveen","front office","office supporter", 25000),

-- (15,"om prakash","back office","", 19000),(16,"parthi","pantry","", 20000),(17,"quereen","","senior supporter", 32000),

-- (18,"revi","IT","data analysist", 55000),(19,"shanmugan","human resources","general manager", 125000),

-- (20,"tanveer","back office","", 23000),(21,"umaiyaal","","", 29000),(22,"vanitha","sales","", 25000),

-- (23,"waheem","IT","software developer", 20000),(24,"xavier","sales","manager", 65000),(25,"yusuf","IT","data analysist", 54000),

-- (26,"zahira","sales","receiptionist", 30000)

-- select \* from officials

-- set sql\_safe\_updates=0

-- update officials set job = "desk supporter" where job="" and name ="om prakash"

-- select \* from officials

-- update officials set job = "incharger" where job=""

-- update officials set department = "help desk" where department = ""

-- update officials set department = "info tech" where department ="IT"

-- select \* from officials

-- select\* from officials where salary is null

-- select \* from officials where name is not null

-- select \* from officials where id in(1,6,9,16,22,19) and salary = 25000

-- select \* from officials where id in(5)

-- select distinct id from officials

-- select count(distinct id) from officials

-- select \* from officials order by salary desc

-- select \* from officials where salary between 0 and 50000

-- select \* from officials where salary >50000

-- select \* from officials where salary between 25000 and 35000

-- select \* from officials

- select \* from officials

-- select \* from officials where department = "sales" and salary > 50000

-- select \* from officials where department ="sales"

-- select \* from officials where department ="help desk"

-- select \* from officials where name like ("m%")

-- select \* from officials where name like ("v%")

-- select \* from officials

-- alter table officials add column age int

-- select \* from officials

-- alter table officials add column gender varchar(10)

-- select \* from officials

-- set sql\_safe\_updates = 0

-- update officials set age ="24" where id ="5"

-- select \* from officials

-- update officials set age ="19" where id =("6")

-- update officials set age = "42" where id="6"

-- update officials set gender = "male"

-- select \* from officials

-- update officials set age = "26" where id = "12"

-- update officials set gender ="female" where id ="26"

-- update officials set gender ="female" where id ="22"

-- update officials set gender ="female" where id ="21"

-- update officials set name ="quereena" where id ="17"

-- update officials set gender ="female" where id ="17"

-- select \* from officials where gender ="female"

-- select \* from officials where gender ="female" and salary >31000

-- select \* from officials

-- update officials set age ="23" where id = ("2")

-- update officials set age ="28" where id = ("1")

-- update officials set age ="34" where id = ("3")

-- update officials set age ="43" where id = ("14")

-- update officials set age ="32" where id = ("9")

-- update officials set age ="29" where id = ("14")

-- select \* from officials

-- update officials set age ="32" where id = ("13")

-- update officials set age ="26" where id = ("15")

-- update officials set age ="30" where id = ("16")

-- update officials set age ="33" where id = ("17")

-- update officials set age ="37" where id = ("18")

-- update officials set age ="41" where id = ("19")

-- update officials set age ="35" where id = ("20")

-- update officials set age ="38" where id = ("21")

-- update officials set age ="28" where id = ("22")

-- update officials set age ="27" where id = ("23")

-- update officials set age ="22" where id = ("24")

-- update officials set age ="46" where id = ("25")

-- select \* from officials

-- update officials set age ="22" where id = ("26")

-- update officials set age ="27" where id = ("17")

-- update officials set age ="16" where id = ("34")

-- update officials set age ="29" where id = ("11")

-- update officials set age ="32" where id = ("10")

-- update officials set age ="25" where id = ("8")

-- update officials set age ="33" where id = ("7")

-- update officials set age ="28" where id = ("4")

select \* from officials

DAY 11 is completed

DAY 12: (12.06.2024)

GROUP BY - which is used to group the rows data as aggregated form which is count all rows depends on we select.

Includes,

Min()

Max()

Avg()

Count()

And

Sum().

WILD CARD – is we can get each name of data’s like we calling name of starting letter.

EDA – Exploratory Data Analysis (Part of Machine Learning)

-- select \* from officials

-- select \* from officials where gender="male"

-- select department, sum(salary) from officials group by department

-- select department, avg(salary) from officials group by department

-- select department, count(\*) from officials group by department

-- select count(\*) from officials

-- select count(name) from officials

-- select department, sum(salary), avg(salary) from officials group by department

-- select sum(salary), avg(salary) from officials group by department

-- select department, sum(salary), avg(age) from officials group by department

-- select \* from officials

-- insert into officials values(28,"punith","sales","sales executive",32500,29,"male")

-- insert into officials value(29,"manimegalai","info tech","SQL developer",36000,26,"female"),(30,"valli","account","cashier",24000,27,"female")

-- select \* from officials

-- select \* from officials where name like("v%"

) show databases

-- use sakila

-- show tables

-- select \* from actor

-- show tables

-- select \* from address

-- select \* from address where address like("m%")

-- select \* from customer\_list

-- select \* from customer\_list where country = "india"

-- select count(\*) from customer\_list where country = "india"

-- select count(\*) from customer\_list where country = "canada"

-- select count(\*) from customer\_list where country = "algeria"

-- use june

-- show tables

-- select \* from officials

-- select sum(salary) from officials group by department

-- select department, sum(salary) from officials group by department

-- select job, sum(salary) from officials group by job

-- select department, avg(salary) from officials group by department

-- select job, avg(salary) from officials group by job

-- select \* from officials

-- select sum(salary) from officials group by department = "info tech"

-- select avg(salary) from officials group by department = "info tech"

-- select sum(salary) as Total\_salary from officials

-- select avg(salary) as average\_salary from officials

-- select \* from officials

-- select sum(salary) from officials where age >30

-- select sum(salary) as above\_thirty from officials where age >30

-- select \* from officials where salary >(select avg(salary)from officials)

-- select \* from officials where name like "p%"

-- select \* from officials where name like "%m"

select \* from officials where name like "%m%"

DAY 12 IS COMPLETED.

DAY 13: (13.06.2024)

show databases

-- use june

-- show tables

-- select \* from officials

-- describe officials

-- select concat(department,job) from officials

-- select concat(department, job) as departjob from officials

-- select concat(department, job) as "call of duty" from officials

-- show databases

-- use june

-- show tables

-- select \* from batch\_std

-- select (english+maths) as "total marks" from batch\_std

-- select (english+maths+science) as "Total marks" from batch\_std

-- select name, (english+maths+science) as "Total marks" from batch\_std

-- select \* from batch\_std

-- select name, (english+maths+science) as "Total marks" from batch\_std where id =5

-- select name, (english+maths+science) as "total marks" from batch\_std where name like "rajan"

-- select \* from batch\_std

-- select count(distinct id) from batch\_std

-- select count(distinct id) as "total ids" from batch\_std

-- select \* from batch\_std

-- select \* from batch\_std order by maths ="36"

-- select \* from batch\_std where maths ="36"

-- select \* from batch\_std where maths ="75" and science ="40" (it wont work because of you given as to know by two rows)

-- select \* from batch\_std

-- select sum(maths) from batch\_std

-- select sum(maths) as "maths total" from batch\_std

-- select sum(science) from batch\_std

-- select \* from batch\_std

DAY 13 IS COMPLETED.

DAY 14: (14.06.2024)

-- use june

-- show tables

-- show databases

-- use fromseppract

-- show tables

-- select \* from person1

-- select \* from movies

-- select min(collection) from movies

-- select max(collection) from movies

-- select max(collection), mov\_name from movies group by mov\_name

-- select mov\_name, min(collection) from movies group by mov\_name

-- select \* from movies where collection >(select avg(collection) from movies)

-- select avg(collection) from movies

-- select \* from movies where collection <(select avg(collection) from movies)

-- select count(mov\_id) from movies

-- select \* from movies where hero\_name like "vijay"

-- select min(collection) from movies where hero\_name like "vijay"

-- select min(collection) as "lowest collection" from movies where hero\_name like "vijay"

-- select \* from movies

-- select \* from movies

-- select mov\_name, min(collection) from movies where mov\_name like "vijay" group by mov\_name

-- select \* from movies where hero\_name like "vijay"

-- set sql\_safe\_updates=0

-- update movies set mov\_name ="mersal", collection =3500000, heroine\_name ="kaajal agarwal" where mov\_id = 3

-- select \* from movies where hero\_name like "vijay"

-- select min(collection),hero\_name from movies group by hero\_name

-- select max(collection),hero\_name from movies group by hero\_name

-- show databases

-- use june

-- show tables

-- select \* from friends

-- select \* from friends where frnd\_salary > 35000

-- select \* from friends where frnd\_job ="java developer"

-- select \* from officials

-- select min(salary), department from officials group by department

-- select department, min(salary) from officials group by department

-- select department, max(salary) from officials group by department

-- select department, avg(salary) from officials group by department

-- (alter table officials add constraint check (salary > 20000)

-- it is about some formula making mistake so it does keep to get an output)

-- select \* from officials

-- select department,age,max(salary) from officials where age between 25 and 35 group by department,age

-- select name,department,age,min(salary) from officials where age between 25 and 35 group by name,department,age

-- select name,department,age,avg(salary) from officials where salary >25000 group by name,department,age

DAY 14 IS COMPLETED.

Day 15 and

DAY 16 is holiday (Saturday, sunday)

DAY 17: (17.06.2024)

DAY 17.06.2024

-- use june

-- show tables

-- create table marks(id int auto\_increment primary key, name varchar(30),

-- tamil int, english int, maths int, science int, social\_sci int )

-- select \* from marks

-- describe marks

-- insert into marks (name, tamil, english, maths, science, social\_sci) values("naveen", 78,45,68,43,56)

-- insert into marks (name, tamil, english, maths, science, social\_sci) values("mady",95,45,69,78,84),("madhan",65,45,52,36,39),("karthi",56,78,64,53,89),

-- ("arul",54,48,37,69,74),("aravindh",45,36,35,84,97),("suresh",78,36,98,62,75),("maadhavi",87,54,65,49,69),("soniya",63,52,47,85,74),

-- ("anbu",85,74,69,65,43),("aanandhi",78,85,96,84,76),("senthil",63,41,52,68,79),("kayalvizhi",65,84,75,84,64),("panneerselvam",89,74,35,58,99)

-- select \* from marks

-- select avg(tamil) from marks

-- select avg(english) from marks

-- select max(maths) from marks

-- select min(science) from marks where id >5

-- select min(science) from marks

-- select \* from marks where maths = 37

-- select name, sum(tamil+english+maths+science+social\_sci) from marks group by name

-- select name,sum(tamil+english+maths+science+social\_sci) as "total marks" from marks group by name

-- select name,concat(tamil,english,maths,science,social\_sci) from marks

-- select sum(maths \*2) as "Twice maths" from marks

-- select sum(maths) from marks

-- select min(english), max(maths), avg(social\_sci) from marks

-- select min(english), max(maths), avg(science) from marks group by name

-- select name, min(english), max(science), avg(social\_sci) from marks group by name

-- select \* from marks

-- select min(tamil), max(maths), avg(science) from marks where name like "m%"

-- select min(tamil), max(maths), avg(science) from marks group by name ="a%"

-- select name, (tamil+english+maths) / 4 as percentage from marks

-- select name, sum(tamil+english+maths+science+social\_sci) from marks group by name

-- select min(tamil), max(science), avg(social\_sci) from marks where name like "k%"

-- select min(english), max(science) from marks where maths=65

DAY 17 IS COMPLETED.

DAY 18: (18.06.2024)

use june

-- show tables

-- select \* from officials

-- select \* from batch\_std

-- select \* from peoples

-- select \* from movie

-- select \*from friends

-- select \* from officials

-- describe friends

-- describe officials

-- alter table friends add primary key(frnd\_id)

-- describe friends

-- alter table officials add primary key(id)

-- describe officials

-- alter table friends add constraint foreign key(frnd\_id) references officials(id)

-- select \* from officials

-- select \* from friends

-- describe friends

-- alter table friends modify column phn\_no int

-- alter table friends add constraint fk\_no foreign key(phn\_no) references officials(id)

-- select \* from friends

-- select officials.department, friends.frnd\_job, officials.salary, friends.frnd\_hometown from officials, friends where

-- officials.id = friends.phn\_no

-- (above, i should find out that how to alter primary key and reference wise

-- once created table already exept by those. (primary & reff)).

-- i tried again but some how got error.

-- create table products(id int primary key auto\_increment, name varchar(20), price int, stock text )

-- create table orders(id int primary key auto\_increment, place text, district text, state text,

-- product\_id int, foreign key(product\_id) references products(id) )

-- select \* from products

-- select \* from orders

-- describe products

-- describe orders

-- insert into products(name,price) values("orange", 30),("apple", 35),("binapple", 20),("mosambi", 40),("watermilon", 25),

-- ("fig", 30),("banana", 30),("papaya", 10),("gouava", 10),("lime", 10),("muskmelon", 20),("orange", 20),("chikku", 20),("graphs", 25),

-- ("amla", 20),("cheetah fruit", 30)

-- select \* from products

-- set sql\_safe\_updates = 0

-- update products set stock = 50 where id = 1

-- update products set stock = 30 where id = 2

-- select count(id) from products

-- select \* from products where id = 9

-- select \* from products

-- update products set stock = 25 where id = 3

-- update products set stock = 36 where id = 4

-- update products set stock = 45 where id = 5

-- select \* from products

-- select \* from orders

-- insert into orders(place,district,state,product\_id) values("coimbatore","coimbatore","tamilnadu", 1)

-- insert into orders(place,district,state) values("kadag","kadag","karnataka")

-- nsert into orders(place,district,state) values("aranthangi","pdukkottai","tamilnadu"),

-- ("kattumavadi","pudukkottai","tamilnadu"),("kk nagar","trichy","trichy"),("thirukogarnam","pudukkottai","tamilnadu"),

-- ("triplecane","chennai","tamilnadu"),("tamparam","chennai","tamilnadu"),("maattuthavani","madurai","tamilnadu"),

-- ("pollachi","coimbatore","tamilnadu"),("pazhani","dindugul","tamilnadu"),("arcode","salem","tamilnadu")

-- select \* from orders

-- select products.name, products.price, orders.district, orders.state from products, orders where products.id = orders.id

-- select \* from orders

-- update orders set state = "tamilnadu" where id = 5

-- select orders.place, products.name, orders.district, products.price from orders, products where orders.id = products.id

-- select \* from products.

Class practice completed but still have some error by to alter as foreign key wise if not exits already.

I got an solution in this. Just because of I have to drop foreign key then only I can try from beginning itself but without foreign key name it does not work cause of I did not put any name of foreign key alone. Straight away I gave as foreign key unless an name.

FINALLY I GOT OUTPUT

-- use june

-- describe drivers

-- show tables

-- describe products

-- describe orders

-- select \* from products

-- select \* from orders

-- select products.name, orders.place from products,orders where products.id = orders.id

-- select \* from officials

-- select \* from friends

-- describe officials

-- describe friends

-- select friends.frnd\_Fname, officials.job, friends.frnd\_salary from officials,friends where officials.id = friends.frnd\_id

FINALLY GOT AN OUTPUT

NEXT DAY -----------🡪

show tables

-- create table buses(id int primary key auto\_increment, b\_name varchar(25), b\_color varchar(25))

-- insert into buses(b\_name,b\_color) values("pollachi xpress","green"),("madurai xpress","dark orange"),("covai xpress","grey"),

-- ("dindugul xpress","red"),("kanchi xpress",""),("pudugai xpress","blue"),("tanjore xpress","white")

-- select \* from buses

-- set sql\_safe\_updates = 0

-- update buses set b\_color ="green" where id =5

-- create table drivers(id int primary key auto\_increment, d\_name varchar(30), phn\_no int, age int, address text, dri\_id int,

-- constraint foreign key(dri\_id) references buses(id))

-- select \* from buses

-- describe buses

-- describe drivers

-- insert into drivers(d\_name, phn\_no, age, dri\_id) values ("marimuthu", 8459635515, 39, 1),("gopal",8769552134, 26 ,2),("raghu",7845391523,35, 3),

-- ("muthu",7813690244, 28, 4),("franklin",8796231546, 23, 5),("balu",9613020487, 32, 6)

-- alter table drivers modify phn\_no text

-- select \* from drivers

-- select \* from buses

-- select buses.b\_name, drivers.d\_name, drivers.phn\_no from buses, drivers where buses.id = drivers.id

-- select buses.b\_name, drivers.d\_name, drivers.phn\_no from buses, drivers

-- use june

-- select \* from buses

-- select \* from drivers

-- select buses.b\_name, drivers.d\_name from buses, drivers

-- select \* from products

-- select \* from orders

-- select products.name, orders.district from products,orders where products.id = orders.id

-- select buses.b\_name, drivers.d\_name from buses, drivers where buses.id = drivers.id

-- set sql\_safe\_updates = 0

-- drop table drivers

-- drop table buses

-- create table buses(id int primary key auto\_increment, b\_name varchar(25), b\_color varchar(25))

-- insert into buses(b\_name,b\_color) values("pollachi xpress","green"),("madurai xpress","dark orange"),("covai xpress","grey"),

-- ("dindugul xpress","red"),("kanchi xpress","green"),("pudugai xpress","blue"),("tanjore xpress","white")

-- select \* from buses

-- create table drivers(id int primary key auto\_increment, d\_name varchar(30), phn\_no int, age int, address text, dri\_id int,

-- foreign key(dri\_id) references buses(id))

-- alter table drivers modify phn\_no text

-- insert into drivers(d\_name, phn\_no, age, dri\_id) values ("marimuthu", 8459635515, 39, 1),("gopal",8769552134, 26 ,2),("raghu",7845391523,35, 3),

-- ("muthu",7813690244, 28, 4),("franklin",8796231546, 23, 5),("balu",9613020487, 32, 6)

-- select \* from drivers

-- i have created table both already which is buses and drivers but earlier i got an id line both table started as contineuing,

-- like where end first table id then second table started from itself where left.

-- so finally, i got an answer for that, which is while creating table and instruct as primary key foreign key as well, in this if i given as

-- "constraint foreign key like then it will take by order both table what id line as continueing."

-- dont give constraint along with foreign key while you create new table with foriegn key and primary key .

-- just give foreign key() like this.

-- select buses.b\_name, drivers.d\_name, drivers.phn\_no from buses,drivers where buses.id = drivers.id

-- select \*from drivers

-- select \*from buses

DAY 19: (19.06.2024)

use june

-- select \* from officials

-- select name from officials where exists(select name from officials where gender = "male")

-- select name from officials

-- select name,salary from officials

-- select name,salary from officials where exists(select name from officials where gender = "male")

-- select name,salary from officials where exists(select name,department from officials where gender = "male")

-- select name,department from officials where gender = "male"

-- select \* from products

-- select \* from orders

-- select name from products where exists(select place from orders where products.id = orders.id)

-- select name from products where exists(select place from orders where products.id = orders.id and orders.id < 3)

-- select name from products where exists(select place from orders where products.id = orders.id and products.id < 5)

-- select \* from officials

-- select \* from officials where salary = any(select salary from officials where age <25)

-- select \* from officials where salary = any(select salary from officials where age <30)

-- select \* from officials where age <25

-- select \* from officials where salary = any(select salary from officials where age <25)

-- select \* from officials where salary = any(select salary from officials where age <24)

-- select \* from officials where salary = any(select salary from officials where salary <30000)

-- select \* from officials where age = any(select age from officials where age <30)

-- select \* from officials where age in(select age from officials where age <30)

-- (in above query ANY , IN both will giev same answe if exists data was correct)

-- select name,job,age from officials where age = any(select age from officials where age <30)

-- select name,job,age from officials where age <30

DAY 20: (20.06.2024)

-- use june

-- select \* from officials

-- select job from officials where exists(select \* from officials)

-- select name from officials where salary=any(select salary from officials where salary >30000)

-- select \* from officials where gender in(select gender from officials)

-- use june

-- show databases

-- select \* from products

-- select \* from orders

-- describe products

-- describe orders

-- select \* from products where name = all(select name from orders where district ="kadag")

-- select name from products where name = all(select name from orders where district ="kadag")

-- select products.name, orders.place, products.stock from products, orders where products.id = orders.id

-- ( in this talk about i have seen in varieties formula which is,

-- exists,

-- any,

-- in,

-- all.

--

-- main part about in this,

-- we can use those as sub query action when its required.

-- though it will consider main query alone when sub query got satiesfied.

-- )

-- select \* from products

-- select \* from orders

-- select \* from products where name =all(select name from orders where state ="karnataka")

-- select \* from products where stock =all(select stock from orders where state ="karnataka")

-- select id, name, price from products where name =all(select name from orders where state ="karnataka")

-- show databases

-- drop database temp\_june

-- use june

-- create table temp\_officials as select \* from officials

-- select \* from temp\_officials

-- drop table temp\_officials

-- describe officials

-- create table temp\_officials(id int, name varchar(30), department varchar(30), job varchar(30), salary int, age int, gender varchar(10))

-- insert into temp\_officials select \* from officials

-- select \* from temp\_officials

-- drop table temp\_officials

-- create table temp\_officials as select id, name, department from officials

-- select \* from temp\_officials

-- drop table temp\_officials

-- create table temp\_officials(id int, name varchar(30), department varchar(30), job varchar(30), salary int, age int, gender varchar(10))

-- insert into temp\_officials select id, name, department, job from officials

-- (it wont work because columns does not match)

-- create table temp\_june.temp\_officials like june.officials

-- select \* from temp\_officials

-- insert into temp\_june.temp\_officials select \* fro

-- drop table temp\_officials

-- create database temp\_june

-- use temp\_june

m june.officials

-- select \* from temp\_officials

-- drop table temp\_officials

-- create table temp\_june.temp\_officials like june.officials

-- insert into temp\_june.temp\_officials select \* from june.officials where age >30

-- select \* from temp\_officials

-- drop table temp\_officials

-- create table temp\_june.temp\_officials like june.officials

-- insert into temp\_june.temp\_officials select \* from june.officials where department = "sales"

-- select \* from temp\_officials

-- drop table temp\_officials

-- create table temp\_june.temp\_officials like june.officials

-- insert into temp\_june.temp\_officials select \* from june.officials where salary >100000

-- select \* from temp\_officials

-- drop table temp\_officials

-- create table temp\_june.temp\_officials like june.officials

-- insert into temp\_june.temp\_officials select name,department, salary from june.officials where salary >100000

-- (it wont work because column s does not matched)

-- drop table temp\_officials

-- drop database temp\_june

-- use june

-- select \* from officials

-- show databases

-- GOT TODAY CLASS ABOUT MUCH

DAY 21: (24.06.2024)

-- (INBUILD STRING FUNCTIONS)

-- select ascii("n") as ascii\_value

-- select ascii("a") as a\_value

-- select ascii("b") as b\_value

-- select ascii("c") as c\_value

-- select ascii("d")

-- select ascii("e") as e\_value

-- select ascii("f") as f\_value

-- select ascii("g") as g\_value

-- select ascii("h") as h\_value

-- select ascii("i") as i\_value

-- select ascii("j")

-- select ascii("k")

-- select ascii("l")

-- select ascii("m")

-- select ascii("n")

-- select ascii('a')

-- select ascii("b")

-- select ascii("c")

-- select ascii("n")

-- select ascii("s")

-- select ascii("t")

-- select ascii("w")

-- select ascii("v")

-- select ascii("x")

-- select ascii("n")

-- select char(65) as character\_c

-- select char(85)

-- select char(65)

-- select char(99) as char\_c

-- select char(100) as char\_v

-- select concat('hello', ' ', 'world')

-- select concat("hi", " ", "marudhu") as maru

-- select concat(54+87)

-- select concat(2\*89)

-- select concat(2\*58/1)

-- select concat(2\*58-10)

-- select sum(110)

-- select concat(23+25+89\*2)

-- select concat(2+4\*3-2+90)

-- select concat(6\*6+4-39+99)

-- select sum(50+50)

-- select sum(2\*2+6)

-- select concat(50+50\*10)

-- select concat(2\*2+6\*9)

-- select concat(2+4\*3-2+90)

-- select concat(3\*5\*2+20)

-- select concat(50+50\*10)

-- select concat(25+25\*2+2)

-- select concat(2+50\*2+20)

-- select concat(2+50+20\*2)

-- select concat(2\*5+9-5)

-- select concat(2+6\*5+9)

-- select concat(2+6+8)

-- select sum(2+6+8)

-- select concat("hi" ,"maru") returns without gaps between

-- select concat("hi",' ', "marudhu") returns with gaps between

-- select instr("hello world","world") returns the posistion

-- select instr("hello world", "orld")

-- select instr("hi hello and how are you ", "and how are you")

-- select length("hello") returns total letters count

-- select length("heloo marudhu") as maru

-- select upper("hello") returns as capitel letters

-- select lower("hi Maru chelLaiah" ) as che returns as small letters

-- select lpad("5","9","8") returns padded left

-- select lpad("3","8","9")

-- select lpad("2","4","5")

-- select lpad("1","8","4")

-- select rpad("6","2","8") returns padded right

-- select rpad("7","3","5")

-- select rpad("3","5","8")

-- select ltrim(" hello") returns as trimmed position

-- select ltrim("he hellojee how are you ")

-- select rtrim(" hi babie how are you") returns as trimmed position

-- select replace("hi hello","hello"," heloo mysql")

-- select replace("marudhu eppadi"," eppadi"," irukinga") returns marudhu irukinga

-- select replace("nalla irukingala","irukingala","maru") as maruche

-- select substring("hello marudhu","8","9") as maru

-- select substring("ghdvgukgjebjeb","5","8") returns from 5th letter

-- select ascii("a")

-- select left("hello world",8) returns hello wo

-- select right("hello world","3") returns rld

-- select repeat("maru", 2) returns marumaru

-- select repeat("maru ",3) returns maru maru maru

-- select find\_in\_set("m","a,b,c,d,m") returns 5

-- select find\_in\_set("hi","ms,kj,hi,hy") returns 3

-- select field("m","k","m","l","g") returns 2

-- select locate("are","how are you") returns 5

-- select position("maru" in"hello maru") returns 7

-- select position("consistent" in "consistent makes you better daily") returns 1

-- select position("you" in "hi hello you are good") returns 10

-- select substring\_index2("www.example.com",".",3) from dual (it is intract with db so didt understand

-- SELECT REPLACE(REPLACE('hello', 'e', 'i'), 'l', 'p') returns hippo

-- select replace(replace("nallarukiya","a","e"),'i','e') returns nellerukeye

-- select replace(replace(replace(replace(replace('entrepreneur','e','0'),'r','1'),'e','2'),'p','3'),'n','5') returns 05t1031050u1

-- select replace(replace(replace('makkale','m','1'),'a','2'),'l','5') returns 12kk25e

JOINS TABLE:

Left, right, inner join

- create database joins

-- use joins

-- create table products(id int, product\_name varchar(30), price int)

-- insert into products values(1,'chair',250)

-- insert into products values(2,'table',300),(3,'pipe',250),(4,'wire',100),(5,'watch',600),

-- (6,'paste',50),(7,'bresh',50)

-- select \* from products

-- create table orders(id int, order\_name varchar(30), quantity int)

-- insert into orders values(1,'aaa',6),(2,'bbb',8),(3,'ccc',7),(4,'eee',11),(5,'ddd',14),(6,'eee',18),(7,'fff',25)

-- select \* from orders

-- select \* from products left join orders on products.id = orders.id

-- select \* from orders left join products on orders.id = products.id

-- select \* from products right join orders on products.id = orders.id

-- select \* from orders right join products on orders.id = products.id

-- select \* from orders left join products on orders.id = products.id

-- select \* from orders right join products on orders.id = products.id

-- select \* from products left join orders on products.id = orders.id

-- union

-- select \* from products right join orders on products.id = orders.id

-- insert into orders values(8,'ggg',20),(9,'hhh',80),(10,'iiii',29)

-- select \* from products left join orders on products.id = orders.id

-- select \* from products right join orders on products.id = orders.id

-- select \* from orders right join products on orders.id = products.id

-- select \* from orders left join products on orders.id = products.id

-- select \* from products right join orders on products.id = orders.id

-- select \* from products left join orders on products.id = orders.id

-- union

-- select \* from products right join orders on products.id = orders.id

-- select \* from products inner join orders on products.id = orders.id

-- select \* from products join orders on products.id = orders.id

-- select \* from products left join orders on products.id = orders.id

-- union

-- select \* from products right join orders on products.id=orders.id

-- select \* from orders left join products on orders.id = products.id

-- select \* from products right join orders on products.id = orders.id

Completed.

-- create table temp\_prod as select \* from products

-- select \* from temp\_prod

-- drop table temp\_prod

-- create table temp as select id,price from products

-- select \* from temp

-- drop table temp

-- create database tejoins

-- use tejoins

-- create table tejoins.tetemp like joins.products

-- select \* from tetemp

-- drop table tetemp

-- drop database tejoins

DAY 22: (25.06.2024)

-- show databases

-- use joins

-- create table products(id int, product\_name varchar(30), price int)

-- insert into products values(1,'chair',250)

-- insert into products values(2,'table',300),(3,'pipe',250),(4,'wire',100),(5,'watch',600),

-- (6,'paste',50),(7,'bresh',50)

-- select \* from products

-- insert into products values(8,'soap',40),

-- (9,'charger',350),

-- (10,'smart phone',19999),

-- (11,'bedsheet',499)

-- select \* from products

-- select max(price),product\_name from products group by product\_name having count(id) <8

-- select max(price), product\_name from products group by product\_name

-- select max(price), product\_name from products group by product\_name having count(id) <20

-- select count(price), product\_name from products group by product\_name

-- select count(price),product\_name,id from products group by product\_name,id having count(id) <5

-- select product\_name,

-- case

-- when price>300 then 'expensive products'

-- when price >500 and 1000 then 'toprated products'

-- else 'highest price product'

-- end as product\_category from products

-- select \* from products

-- select product\_name,

-- case

-- when price > 10 and 100 then 'medium price product'

-- when price > 100 and 1000 then 'expensive product'

-- else 'highrated price' end as product\_wise

-- from products

-- select product\_name,

-- case

-- when price >1000 then 'highrated price'

-- when price >500 then 'expensive product'

-- else 'medium rated price' end as product\_wise

-- from products

-- insert into products values(null,'mug',399),(13,null,2000),(14,'ladder',null),

-- (15,'pillows',499),(16,'iron box',2500)

-- select \* from products

-- select product\_name, price \*(id + ifnull(id,0)) from products; (not understanding yet this query) but understand part of

-- null function having reason

DAY 23: (26.06.2024)

-- use joins

-- select ascii('a')

-- select chr(65) from dual (not getting)

-- select "hello" || "" || "world" as

-- select "hi" || "" || "marudhu"

-- select dump("hello",2) as abc (not getting)

-- select initcap("hello world") (not getting)

-- select length('hello')

-- select lower('hello')

-- select upper('hello')

-- select ltrim('hello')

-- select rtrim('hello')

-- select replace('hello world','world','mysql')

-- select reverse('hello')

-- select left('hello world',5)

-- select right('hello world',5)

-- select repeat('hello',9)

-- select locate('maru', 'he is from IT department and name of maru')

-- select abs(-9)

-- select abs(+9)

-- select abs(-8)

-- select abs(-9)

-- select ceil(5.2)

-- select ceil(6.8) returns round off next number

-- select exp(1)

-- select exp(2)

-- select exp(3)

-- select exp(-3)

-- select floor(9)

-- select floor(8)

-- select greatest(25,54,78,23,456,487)

-- select least(41,52,63,25,14,36)

-- select least(10,20,30,40,50) as man

-- select mod(5,8)

-- select mod(8,5)

-- select mod(2,6)

-- select mod(6,2)

-- select power(20,22)

-- select power(2,5)

-- select power(3,6)

-- select power(5,9)

-- select power(9,12) returns by multiple

-- select round(2.35487,7)

-- select round(6.2451,9) returns as 6.2451

-- select sqrt(64)

-- select sqrt(81) returns as dividing

-- select sysdate()

-- select sysdate(6) returns as individual lenth number

-- select current\_timestamp()

-- select extract(year from current\_date())

-- select extract(year from current\_date()) as current\_year

-- select extract(month from current\_date)

-- select extract(day from current\_date())

-- select timestampdiff(month,"2023-08-20","3034-06-24")

-- select timestampdiff(year,'2022-12-25','2025-01-20')

-- select cast(99 as char)

-- select cast(85 as char) returns 85

-- select convert(12345,char)

-- select convert(254687,char) returns 254687

DAY 24: (27.06.2024)

- login as root

-- mysql -u root -p

-- grand necessary privilages to your user

-- grant create user on \*.\* to 'your-current-user'@'your\_host'

-- grant grant option on. to 'your-current-user'@'your\_host' (tips)

-- login as root

-- mysql -u root -p

-- grand necessary privilages to your user

-- grant create user on \*.\* to 'your-current-user'@'your\_host'

-- grant grant option on. to 'your-current-user'@'your\_host' (tips)

-- use joins

-- show tables

-- mysql -u root -p

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'maru'@'localhost' identified by 'password'

-- grant select, insert on joins.\* to 'maru'@'localhost'

-- revoke select,insert on joins.\* from 'maru'@'localhost'

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'che'@'localhost' identified by 'password'

-- grant select,insert on joins.\* to 'che'@'localhost'

-- revoke select,insert on joins.\* from 'che'@'localhost'

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'vel'@'localhost' identified by 'password'

-- grant select,insert on joins.\* to 'vel'@'localhost'

-- revoke select,insert on joins.\* from 'vel'@'localhost'

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'pan'@'localhost' identified by 'password'

-- grant select,insert,alter on joins.\* to 'pan'@'localhost'

-- revoke select,insert,alter on joins.\* from 'pan'@'localhost'

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'veche'@'localhost' identified by 'password'

-- grant select,insert,alter,delete on joins.\* to 'veche'@'localhost'

-- revoke select,insert,alter,delete on joins.\* from 'veche'@'localhost'

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'don'@'localhost' identified by 'password'

-- grant select, insert, delete, alter on joins.\* to 'don'@'localhost'

-- then

-- revoke select, insert, delete, alter on joins.\* from 'don'@'localhost'

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'maddy'@'localhost' identified by 'password'

-- grant select, alter, insert on joins.\* to 'maddy'@'localhost'

-- then

-- revoke select,alter, insert on joins.\* from 'maddy'@'localhost'

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'love'@'localhost'

-- grant select,alter on joins.\* to 'love'@'localhost'

-- then

-- revoke select, alter on joins.\* from 'love'@'localhost'

-- show databases

-- grant create user on \*.\* to 'root'@'localhost'

-- create user 'maruu'@'localhost' identified by 'password'

-- grant select, insert, alter on fromseppract.\* to 'maruu'@'localhost'

-- then

-- revoke select, insert, alter on fromseppract.\* from 'maruu'@'localhost'

-- rank - used to assign rank of each rows as a part of result

-- use joins

-- show tables

-- create table employees(

-- id int,

-- name varchar(30),

-- department varchar(30),

-- salary int

-- )

-- select \* from employees

-- alter table employees modify salary float

-- describe employees

-- insert into employees values

-- (1, "Alice", "HR", 50000),

-- (2, "Bob", "HR", 60000),

-- (3, "Charlie", "IT", 80000),

-- (4, "david", "IT", 60000),

-- (5, "Eve", "IT", 70000)

-- select \* from employees

-- select

-- id,

-- name,

-- department,

-- salary, rank()over(partition by department order by salary desc) as rank\_wise from employees

-- rank - is used to rank the every individual rows as part of results.

-- oncemore

-- select

-- name,

-- department,

-- rank()over(partition by department order by salary desc) as rankie from employees

-- additinal of rank funtion is dense\_rank

-- select

-- id,

-- name,

-- department,

-- salary,

-- dense\_rank()over(partition by department order by salary desc) as rank\_wise

-- from employees

DAY 25: (28.06.2024)

-- show databases

-- use fromseppract

-- show tables

-- use joins

-- show tables

-- INDEX - is the database objects that

-- improves the speed of data retrivel operation in the table.alter

-- create index idx\_emp\_salary on employees(salary)

-- select \* from employees

-- show index from employees

-- select \* from employees where salary = 50000

-- drop index idx\_emp\_salary on employees

-- show index from employees

-- composite index - we can index on more than one coloumn

-- create index idx\_emp\_salary on employees(name, salary)

-- show index from employees

-- drop index idx\_emp\_salary on employees

-- TCL (Transaction Control Language)

-- (commit,

-- rollback,

-- savepoint)

-- commit - is we can add more than columns

-- rollback- is it will do the undo transaction if transaction not yet been commited

-- savepoint - is we can store the amount of transaction while do the changes for rollback

-- (rollback - is used to undo the transaction that not yet been

-- commited into the databases)

-- commits used to save all changes made during the current data migration or transaction

-- select \* from employees

-- COMMIT

-- start transaction;

-- insert into employees(id,name,department,salary) values (10,'gayele','accounts',40000),

-- (11,'ameenuth','IT',38000);

-- commit;

-- i got the points regarding commit but some doubts on that rollback. need to shortout once

-- select \* from employees

-- view - is used to create virtual table that will allow us to see the datas more than one table

-- create view hiring\_emp as select id,name,department,salary from employees where department = 'hr'

-- select \* from hiring\_emp

-- show tables

-- select \* from orders

-- create view hiring\_employee as select e.id, e.name, e.department, e.salary, o.order\_name, o.quantity

-- from employees e join orders o on e.id = o.id

-- select \* from hiring\_employee

Completed.

I HAVE COMPLETED OVERALL ABOUT SQL.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL PROJECT:

**Students Databases Management of School**

-- I am going to do the project of SQL.

-- step 1: create the database and tables.

-- create database schoolDB

-- use schoolDB

-- (table for storing student info)

-- create table students(

-- student\_id int primary key,

-- first\_name varchar(50),

-- last\_name varchar(50),

-- date\_of\_birth date,

-- gender char(1)

-- )

-- select \* from students

-- (table for storing course info)

-- create table course(

-- course\_id int primary key,

-- course\_name varchar(100),

-- credits int

-- )

-- (table for storing enrollment info)

-- create table enrollments(

-- enrollment\_id int primary key,

-- student\_id int,

-- course\_id int,

-- enrollment\_date date,

-- grade char(1),

-- foreign key (student\_id) references students(student\_id),

-- foreign key (course\_id) references course(course\_id)

-- )

-- step 2: insert sample data 1. \*insert data into student table\*

-- insert into students(student\_id, first\_name, last\_name, date\_of\_birth, gender)values

-- (1,'john','doe','2000-01-15','M'),

-- (2,'jane','smith','1999-03-22','F'),

-- (3,'mike','johnson','2001-07-30','M'),

-- (4,'emily','davis','2002-10-05','F')

-- (2.insert data into course table)

-- insert into course(course\_id, course\_name, credits) values

-- (1,'mathematics',3),

-- (2,'english',4),

-- (3,'computer science',3),

-- (4,'history',3)

-- (3.insert data into enrollement table)

-- insert into enrollments(enrollment\_id, student\_id, course\_id, enrollment\_date, grade)values

-- (1,1,1,'2023-01-15','A'),

-- (2,2,2,'2023-01-16','B'),

-- (3,1,3,'2023-01-17','A'),

-- (4,3,1,'2023-01-18','C'),

-- (5,4,4,'2023-01-19','B')

-- step 3: write SQL queries. 1. retrive all students

-- select \* from students

-- (2.retrive all courses)

-- select \* from course

-- (3.retrive all enrollments)

-- select \* from enrollments

-- (4.get the names of students enrolled in a specific courses)

-- select students.first\_name, students.last\_name from students

-- join enrollments on students.student\_id = enrollments.student\_id

-- where enrollments.course\_id = 1

-- (5.get the list of courses a specific student is enrolled in)

-- select course.course\_name from course join

-- enrollments on course.course\_id = enrollments.course\_id where

-- enrollments.student\_id = 1

-- -- (6.get the average grade of a specific course)

-- select course.course\_name, avg(case

-- when grade = 'A' then 4.0

-- when grade = 'B' then 3.0

-- when grade = 'C' then 2.0

-- when grade = 'D' then 1.0

-- else 0.0

-- end) as average\_grade from course

-- join enrollments on course.course\_id = enrollments.course\_id

-- where course.course\_id = 1 group by

-- course.course\_name

-- (7. count the number of students in each course)

-- select course.course\_name, count(enrollments.student\_id) as student\_count from course

-- join enrollments on course.course\_id = enrollments.course\_id group by

-- course.course\_name

-- (8. list all students with their enrolled course)

-- select students.first\_name, students.last\_name, course.course\_name from students

-- join enrollments on students.student\_id = enrollments.student\_id join course on

-- course.course\_id = enrollments.course\_id order by students.last\_name, students.first\_name

-- step 4: advanced sql queries \*find students with no enrollments\*

-- select students.first\_name, students.last\_name

-- from students

-- left join enrollments on students.student\_id = enrollments.student\_id where enrollments.student\_id is null

-- (10. get the total number of enrollments per students)

-- select students.first\_name, students.last\_name, count(enrollments.enrollment\_id)

-- as total\_enrollments from students

-- left join enrollments on students.student\_id = enrollments.student\_id group by students.student\_id,

-- students.first\_name, students.last\_name

-- (11.find the highest grade for each course)

-- select course.course\_name, max(enrollments.grade) as highest\_grade

-- from course

-- join enrollments on course.course\_id = enrollments.course\_id

-- group by course.course\_name

-- (12.calculate the average grade per students)

-- select students.first\_name, students.last\_name,

-- avg(case

-- when grade = 'A' then 4.0

-- when grade = 'B' then 3.0

-- when grade = 'C' then 2.0

-- when grade = 'D' then 1.0

-- else 0.0

-- end) as average\_grade from students

-- join enrollments on students.student\_id = enrollments.student\_id

-- group by students.first\_name, students.last\_name

-- (13. identify courses with more than 2 students enrolled)

-- select course.course\_name, count(enrollments.student\_id) as student\_count

-- from course

-- join enrollments on course.course\_id = enrollments.course\_id

-- group by course.course\_name

-- having count(enrollments.student\_id) > 2

-- (14. list students and their grades for a specific course)

-- select students.first\_name, students.last\_name, enrollments.grade

-- from students

-- join enrollments on students.student\_id = enrollments.student\_id

-- where enrollments.course\_id = 1 (replcae with the course\_id you are intrested in)

-- (15. get enrollments details for students born after a specific date)

-- select students.first\_name, students.last\_name, enrollments.enrollment\_date, course.course\_name

-- from students

-- join enrollments on students.student\_id = enrollments.student\_id

-- join course on enrollments.course\_id = course.course\_id

-- where students.date\_of\_birth > '2000-01-01' (replace with rhw date you are intrested in)

-- step 5: adding new features (16.\* addatable for teachers \*)

-- create table teachers(

-- teacher\_id int primary key,

-- first\_name varchar(50),

-- last\_name varchar(50),

-- hire\_date date)

-- (17. link courses to teachers)

-- alter table course

-- add teacher\_id int,

-- add foreign key (teacher\_id) references teachers(teacher\_id)

-- 18. insert sample data into teachers table

-- insert into teachers(teacher\_id, first\_name, last\_name, hire\_date) values

-- (1, 'alice', 'brown', '2015-08-01'),

-- (2, 'bob', 'green', '2018-01-15')

-- update course set teacher\_id = 1 where course\_id in(1,2) (assigned mathematics,english)

-- update course set teacher\_id = 2 where course\_id in(3,4) (assigned computer science, history)

-- step 6: (more complex queries)

-- (19. find courses taught by each teacher)

-- select teachers.first\_name as teacher\_first\_name,

-- teachers.last\_name as teacher\_last\_name, course.course\_name

-- from teachers

-- join course on teachers.teacher\_id = course.teacher\_id

-- (20. calculate average grade given by each teacher)

-- select teachers.first\_name, teachers.last\_name,

-- avg(case

-- when grade = 'A' then 4.0

-- when grade = 'B' then 3.0

-- when grade = 'C' then 2.0

-- when grade = 'D' then 1.0

-- else 0.0

-- end) as average\_grade from teachers

-- join course on teachers.teacher\_id = course.teacher\_id

-- join enrollments on course.course\_id = enrollments.course\_id

-- group by teachers.first\_name, teachers.last\_name

-- (21. list teachers and the number of courses they teach)

-- select teachers.first\_name, teachers.last\_name, count(course.course\_id) as course\_count

-- from teachers

-- left join course on teachers.teacher\_id = course.teacher\_id

-- group by teachers.first\_name, teachers.last\_name