SQl P2

Saturday, August 12, 2023 8:08 PM

ONE TO ONE

```
CREATE TABLE `user` (`id` INT PRIMARY KEY AUTO_INCREMENT,`name` VARCHAR(90),`phone`
VARCHAR(13) ,`ssn` INT);

CREATE TABLE `ssn` (`id` BIGINT PRIMARY KEY AUTO_INCREMENT,`first_name`
VARCHAR(30),`second_name` VARCHAR(30),`third_name` VARCHAR(30),`adddress` VARCHAR(100)
,`status` BOOLEAN,`job` VARCHAR(100));
```

ONE TO MANY

```
CREATE TABLE `category` (`id` INT PRIMARY KEY AUTO_INCREMENT ,`name` VARCHAR(100));
CREATE TABLE `products` (ID INT PRIMARY KEY AUTO_INCREMENT,name VARCHAR(70),price FLOAT
);

ALTER TABLE `products` ADD `category` INT NOT NULL AFTER `price`, ADD INDEX
(`category`);

CREATE TABLE `order` (`id` INT PRIMARY KEY AUTO_INCREMENT,`order-desc` TEXT ,`cst`
VARCHAR(100) );

INSERT INTO `order` (`id`, `order-desc`, `cst`) VALUES (NULL, 'ahmed', 'test'), (NULL,
'muhammed', 'test');

CREATE TABLE `product_order` (`id` INT PRIMARY KEY AUTO_INCREMENT,`product_id` INT
,`order_id` INT );

ALTER TABLE `product_order` ADD INDEX(`product_id`);

ALTER TABLE `product_order` ADD INDEX(`order_id`);
```

Udemy DataBase

Enities: Student , Courses , Lessons , User

```
CREATE TABLE `student` (`id` INT PRIMARY KEY AUTO_INCREMENT ,`firstname` VARCHAR(100)
,`lastname` VARCHAR(100),`bd` DATE ,`username` VARCHAR(100),`email`
VARCHAR(100),`password` VARCHAR(100));

CREATE TABLE `user` (`id` INT PRIMARY KEY AUTO_INCREMENT,`firstname` VARCHAR(100),`lastname`
VARCHAR(100),`email` VARCHAR(100),`password` VARCHAR(100),`usertype` VARCHAR(100));

CREATE TABLE `course` (`id` INT PRIMARY KEY AUTO_INCREMENT,`name` VARCHAR(100),`start`
DATE ,`end` DATE,`duration` INT);

CREATE TABLE `course` (`id` INT PRIMARY KEY AUTO_INCREMENT,`name` VARCHAR(100),`start`
DATE ,`end` DATE,`duration` INT);

CREATE TABLE `usertype` (`id` INT PRIMARY KEY AUTO_INCREMENT,`title` VARCHAR(100));

INSERT INTO `usertype` (`id`, `title`) VALUES (NULL, 'admin'), (NULL, 'instructor');
```

JOIN

-inner join == ||

```
SELECT * FROM `enroll` INNER JOIN student ON enroll.student_id=student.id;

SELECT enroll.course_id ,enroll.date,student.firstname FROM `enroll` INNER JOIN student
ON enroll.student_id=student.id;

SELECT enroll.date,student.firstname ,student.lastname,course.name FROM `enroll` INNER JOIN student ON
enroll.student_id=student.id INNER JOIN course ON enroll.course_id=course.id;

SELECT * FROM `order` RIGHT JOIN product_order ON product_order.order_id=`order`.id;

SELECT student.firstname as fname,course.name as coursename FROM `enroll` INNER JOIN `student` ON
enroll.student_id=student.id INNER JOIN `course` On enroll.course_id=course.id;
```

Functions SQL

```
SELECT *,MIN(price) as `min_price` FROM `products`;

SELECT MAX(price) as `max_price` FROM `products`;

SELECT SUM(price) as `sum_price` FROM `products`;
SELECT AVG(price) as `AVERAGE` FROM `products`;
SELECT TOP 2 * FROM `products`;// sql server ,MS Access
SELECT * FROM `products` LIMIT 2; //Mysql
SELECT * FROM `products` FETCH FIRST 3 ROWS ONLY;
SELECT * FROM `products` WHERE `category` IN (1,5);
SELECT * FROM `products` WHERE `category` = 1 OR `category` =2;
SELECT * FROM `products` WHERE `category` IN(SELECT COUNT(products.id) FROM products);
//complex query
SELECT * FROM `products` WHERE `category` IN(SELECT COUNT(products.id)-4 FROM products);
```
//complex query
```
SELECT * FROM `products` WHERE `category` NOT IN(1,2);
SELECT * FROM `products` WHERE name like 's%';
SELECT * FROM `products` WHERE name like '%g';
SELECT * FROM `products` WHERE name like "_u%";
SELECT * FROM `products` WHERE name like "%su%";

SELECT * FROM `products` UNION SELECT * FROM `user`;
```
//same length of columns not rows -Note-

group by

Here we use HAVING instead of WHERE to make condition

```
SELECT review.id as review_id , student.firstname as stud_name , course.name as
course_name FROM `review` INNER JOIN `student` ON review.student_id=student.id INNER JOIN
`course` ON review.course_id=course.id;
```

SELECT review.id as review_id , student.firstname as stud_name , course.name as course_name FROM `review` INNER JOIN `student` ON review.student_id=student.id INNER JOIN `course` ON review.course_id=course.id GROUP BY course.name;

SELECT review.id as review_id , student.firstname as stud_name , course.name as course_name,AVG(review.review_rate) as rev_rate FROM `review` INNER JOIN `student` ON review.student_id=student.id INNER JOIN `course` ON review.course_id=course.id GROUP BY course.name;

CREATE VIEW `review_rate` AS SELECT review.id as review_id , student.firstname as stud_name , course.name as course_name,AVG(review.review_rate) as rev_rate FROM `review` INNER JOIN `student` ON review.student_id=student.id INNER JOIN `course` ON review.course_id=course.id GROUP BY course.name;

select `udemy`.`review`.`id` AS `review_id`,`udemy`.`student`.`firstname` AS `stud_name`,`udemy`.`course`.`name` AS `course_name`,avg(`udemy`.`review`.`review_rate`) AS `rev_rate` from ((`udemy`.`review` join `udemy`.`student` on(`udemy`.`review`.`student_id` = `udemy`.`student`.`id`)) join `udemy`.`course` on(`udemy`.`review`.`course_id` = `udemy`.`course`.`id`)) group by `udemy`.`course`.`name` having avg(`udemy`.`review`.`review_rate`) >3

Stored Procedures

(Like Function and use CALL to call it)
Udemy -> routine->procedure_name->query

```
CREATE PROCEDURE `Course_Search`(IN `keyword` VARCHAR(100)) NOT DETERMINISTIC CONTAINS
SQL SQL SECURITY DEFINER SELECT * FROM `course` WHERE `name` LIKE `%keyword%`;


SELECT * FROM `course` WHERE `name` LIKE CONCAT('%',keyword,'%');

SELECT * FROM `course` WHERE `id` = keyword;

CALL Course_Search('b');
```

Trigger (like event in js)

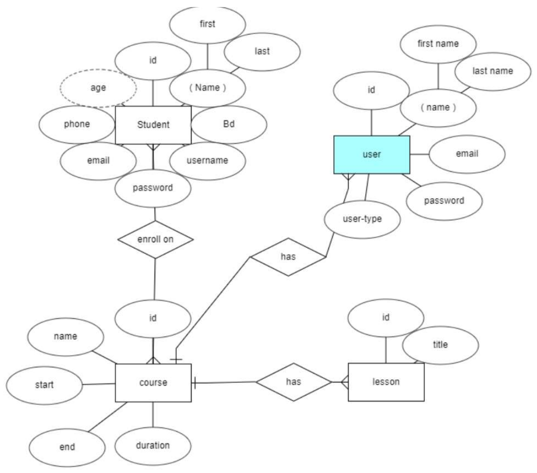
DataBase --> Tiggers -->Tigger Name --On--> table -->Time(after-before)--> Event(Delete-update-insert)--->defination(query)

```
SELECT * FROM `student` ORDER By `firstname` DESC;


SELECT * FROM `course` WHERE start BETWEEN '2020-01-01' AND '2020-03-03';


SELECT * FROM `course` WHERE start BETWEEN '2020-01-01' AND '2020-03-03' AND end BETWEEN
'2021-01-01' AND '2022-01-01';
```

Udemy Erd


first
last
id
age
(Name)
phone
Student
Bd
email
username
password
enroll on
first name
last name
id
(name)
user
email
password
user-type
has
id
name
start
course
end
duration
has
id
title
lesson