**CIC Practicum Monitoring System**

**Analysis, Design and Implementation**

**June 20,2020**

**BSIT1-A**

**Leah C. Juarez**

**Table Of Contents**

**I EXECUTIVE SUMMARY 3**

**II REQUIREMENTS SPECIFICATION 6**

2.1 Existing Information Systems 6

2.2 The Proposed System 7

2.3 Scope 10

**III CONCEPTUAL DATABASE DESIGN 11**

3.1 Identified User Views 11

3.2 Conceptual Data Model 16

**IV LOGICAL DATABASE DESIGN 18**

**V PHYSICAL DATABASE DESIGN 22**

5.1 Physical Data Model 22

5.2 Final Data Dictionary 24

5.3 Final Normalized Tables 28

5.4 Integrity Contraints 29

5.5 Triggering Operations 30

5.6 Views 31

5.7 Stored Procedures 37

**CHAPTER I**

**EXECUTIVE SUMMARY**

A quality education produces graduates equip with knowledge and skills necessary to excel in their chosen field or future careers. That’s why University programs incorporated internship in their curriculum to help students not only to know and memorize but to fully understand the lessons and skills they ought to know and must have. Internship is also called practicum or work placement programs. This program is designed to provide students with practical work experience They emphasize the importance of learning by doing. They’re where students can transfer their knowledge to actual work (canscribe.com, 2017 ) With this great program comes with the great responsibility or workload for instructors .The students are oblige to submit their report after the needed number of hours is completed or by the end of the semester. Considering the number of student’s who will pass their report, handling a large number of students seems to be difficult to monitor. In light of this problem a practicum management system is created.

CIC PMS (College of Institute and Computing Practicum Monitoring System) is a web-based system built to help students and instructors to easily access, store, retrieve practicum files and records which help students submit their end semester report to their instructors without having trouble going to the instructor’s office from their initial location and vice-versa. Students will have their own user accounts for user identification in the system thereby giving them the capability to upload their own files and have access to other data or information related or intended for them. Similarly, the practicum coordinator or the practicum instructor of the college will be given their own accounts to access, download and check if the students already submitted their end semester report. The system will let instructors easily monitor their students and can look through all other or old practicum records, assuming that the request to look up is granted by person in authority.

The system has 3 levels of user access. The admin, instructor and student (mentioned starting with the highest user privilege respectively) .The admin level can look through all data and at the same time is responsible in populating the database with new students and instructor . The admin is the only one who can create new accounts for the new instructor and students so that nobody can just create an account which may result to duplication or waste of database space because the person who created the account is not an intern. There are only few restrictions of the admin which is some of the functions of instructor and students have like changing the status of a student report to “checked” is not present and uploading, deleting a file. The only thing an admin can delete is a record of section-history—A record in which tells you who is the instructor of that section and in what year. Next, is instructor, an instructor user level can only see the practicum records of his/her students. They can view, download and change the file status to “checked” or “uncheck” of their students. Lastly, the student user level, the only information they can access is their record data, their file report and the list of companies. They can upload, view, delete their file and check out if the file they submitted is already checked. The list of companies also has hiring description so that if they are still finding a company to apply for internship they can just scroll to company description on the screen.

Many schools and students already have made and proposed practicum monitoring systems to solve this common problem face by the instructors. One of them has entitled their study as “WEB BASED INTERNSHIP MANAGEMENT SYSTEM,A COLLABORATIVE COORDINATING TOOL ”. Using existing system as basis and the knowledge learned from the 3 programs Information Management ,Advance Information Management and Application Systems Development, the developer consider the system built to be useful and with some guidance and improvements it can be a potential part and become a necessity or the main tool in monitoring student practicum records and requirements. The system is created and design only for students and staff of the University of Southeastern Philippines under the College of Institute and Computing.

**CHAPTER II**

**REQUIREMENTS SPECIFICATION**

**2.1 Existing Information Systems**

Web Based Internship Coordinating System (WICS ) is a system created by Vishal Dharod as a requirement for his Master’s Degree in Computer Science. WICS (Web Based Internship Coordinating System) is a comprehensive toolkit for managing and coordinating internship programs in an educational institution. WICS was developed with the intension of providing an interactive tool for the faculty and students to communicate with each other whenever and wherever they want to. Students can understand the requirements and view their progress and interact with the faculty in a better way. An instructor can review and update a student's progress and send email to students directly if necessary from WICS. Designed in PHP (Hypertext Preprocessor), an open source, server-side, HTML (Hypertext Markup Language) embedded scripting language used for creating dynamic web pages, this piece of software, is easy to maintain and further enhancements can be added without any hassles. In the future, a progress report can be made available to the student in Protable Document Format (PDF). This will include a comprehensive summary of the student's progress and mention tasks that are remaining as well. The view progress page for both the faculty and students can be made more graphical and a bar graph or a pie chart can be displayed to show the progress and have the user better interpret the result. WICS can also have an interface for adding more requirement documents into the internship class and assign the administrator the privilege of adding such documents. WICS should also make a note of the time period of the internship for the faculty to judge the number of units it could be worth for. Before the deployment, hardware and security issues need to be reviewed for the system to run successfully. It would also be strongly recommended to have a person maintain this system if deployed in a big production environment. This could be the administrator of the system. (Dharod,2004)

**2.2 The Proposed System**

CIC PMS (College of Institute and Computing Practicum Monitoring System) is a web-based system built to help students and instructors to easily access, store, retrieve practicum files and records which help students submit their end semester report to their instructors without having trouble going to the instructor’s office from their initial location and vice-versa. Students will have their own user accounts for user identification in the system thereby giving them the capability to upload their own files and have access to other data or information related or intended for them. Similarly, the practicum coordinator or the practicum instructor of the college will be given their own accounts to access, download and check if the students already submitted their end semester report. The system will let instructors easily monitor their students and can look through all other or old practicum records, assuming that the request to look up is granted by person in authority.

The objective of the CIC Practicum Monitoring System (CIC PMS) is :

(1.) to technologized the manual labor of monitoring, retrieving files and records relating to practicum or internship.

(2.)The system will let the instructor find the records of the intern students on a few clicks rather than getting it in the shelves or finding it on the pile of papers. It will not be difficult to instructors to monitor large number of students.

.(3.) Help the students submit their reports online which save them more time, effort and money.

The general range of processing or functions that will be handled by the system is to insert, store, view, edit instructor information , student information, student practicum records & intern companies information . The system has 3 levels of user access. The admin, instructor and student (mentioned starting with the highest user privilege respectively) .The admin level can look through all data and at the same time is responsible in populating the database with new students and instructor . The admin is the only one who can create new accounts for the new instructor and students so that nobody can just create an account which may result to duplication or waste of database space because the person who created the account is not an intern. There are only few restrictions of the admin which is some of the functions of instructor and students have like changing the status of a student report to “checked” is not present and uploading, deleting a file. The only thing an admin can delete is a record of section-history—A record in which tells you who is the instructor of that section and in what year. Next, is instructor, an instructor user level can only see the practicum records of his/her students. They can view, download and change the file status to “checked” or “uncheck” of their students. Lastly, the student user level, the only information they can access is their record data, their file report and the list of companies. They can upload, view, delete their file and check out if the file they submitted is already checked. The list of companies also has hiring description so that if they are still finding a company to apply for internship they can just scroll to company description on the screen.

The general range of the number transactions processed will not be on daily basis but on monthly basis. The system is design for CIC college in USEP so the estimation will be based on the College situation. A transaction is made when a user update, insert, delete ,view an information from the system.

Table 1 Range of the number of transactions processed by a user on a monthly basis:

**Month**

**Users**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Jan | Feb | Mar | Apr | May | June | July | Aug | Sept | Oct | Nov | Dec |
| Admin | 800 | 50 | 5 | 5 | 50 | 500 | 50 | 900 | 70 | 5 | 5 | 40 |
| Instructor | 20 | 5 | 5 | 200 | 500 | 50 | 100 | 10 | 5 | 5 | 200 | 500 |
| Student | 50 | 5 | 5 | 5 | 5 | 5 | 5 | 150 | 5 | 5 | 5 | 5 |

The peak load for this system is for every start and end of summer, first and second semester .(Peak load – the times during the day/week when there will be a high demand made on the database.) The administrator will start adding new user accounts to the system and meanwhile, students will be may or may not simultaneously pass file reports while teachers starts to check submitted reports.

.The estimation is that every semester the active user is the following :

-the maximum instructors assigned to supervise the sections are 6 .

-the maximum total number of intern is 100.

- the maximum number of admin is 3.

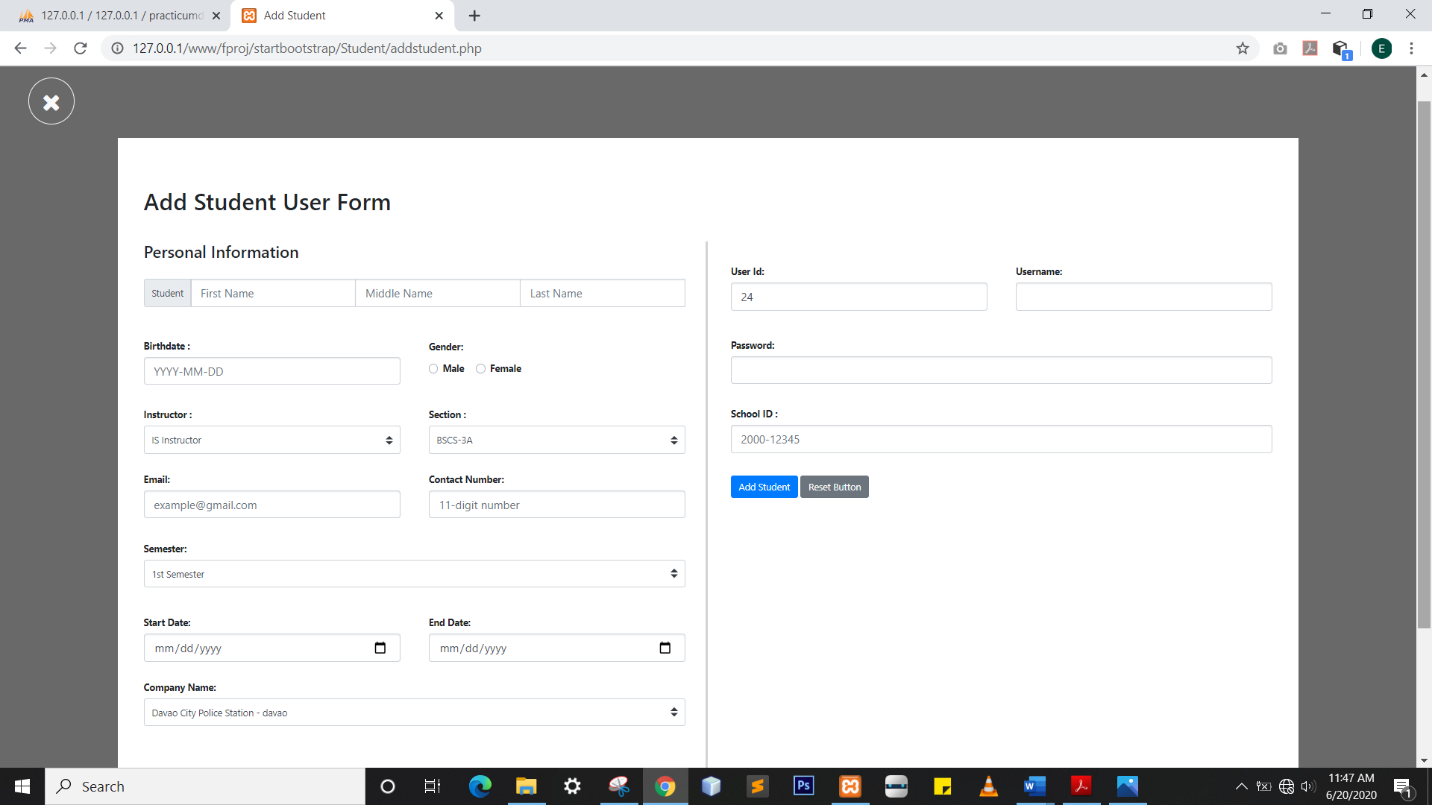
**2.3 Scope**

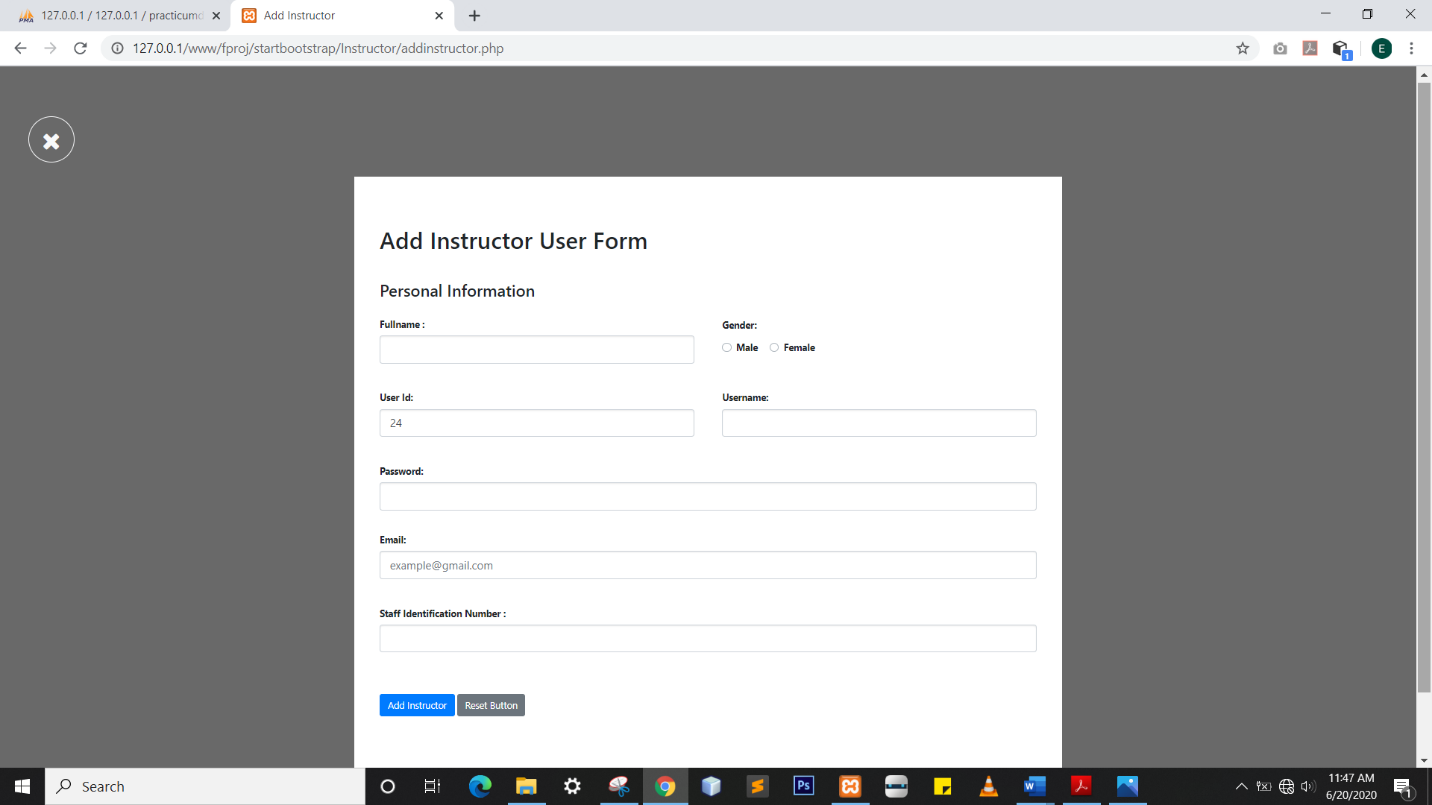
The developed system is for student and staff of USEP-CIC. Therefore, the database is populated based on the CIC existing data. Students with no account cannot access the system . Students who are not enrolled in a practicum program would not be given an account by the administrator. The same as the instructor the instructor would not be given an account if the instructor is not a practicum instructor or coordinator. The users views in this system are the admin, instructor and student.

**CHAPTER III**

**CONCEPTUAL DATABASE DESIGN**

**3.1 Identified User Views**

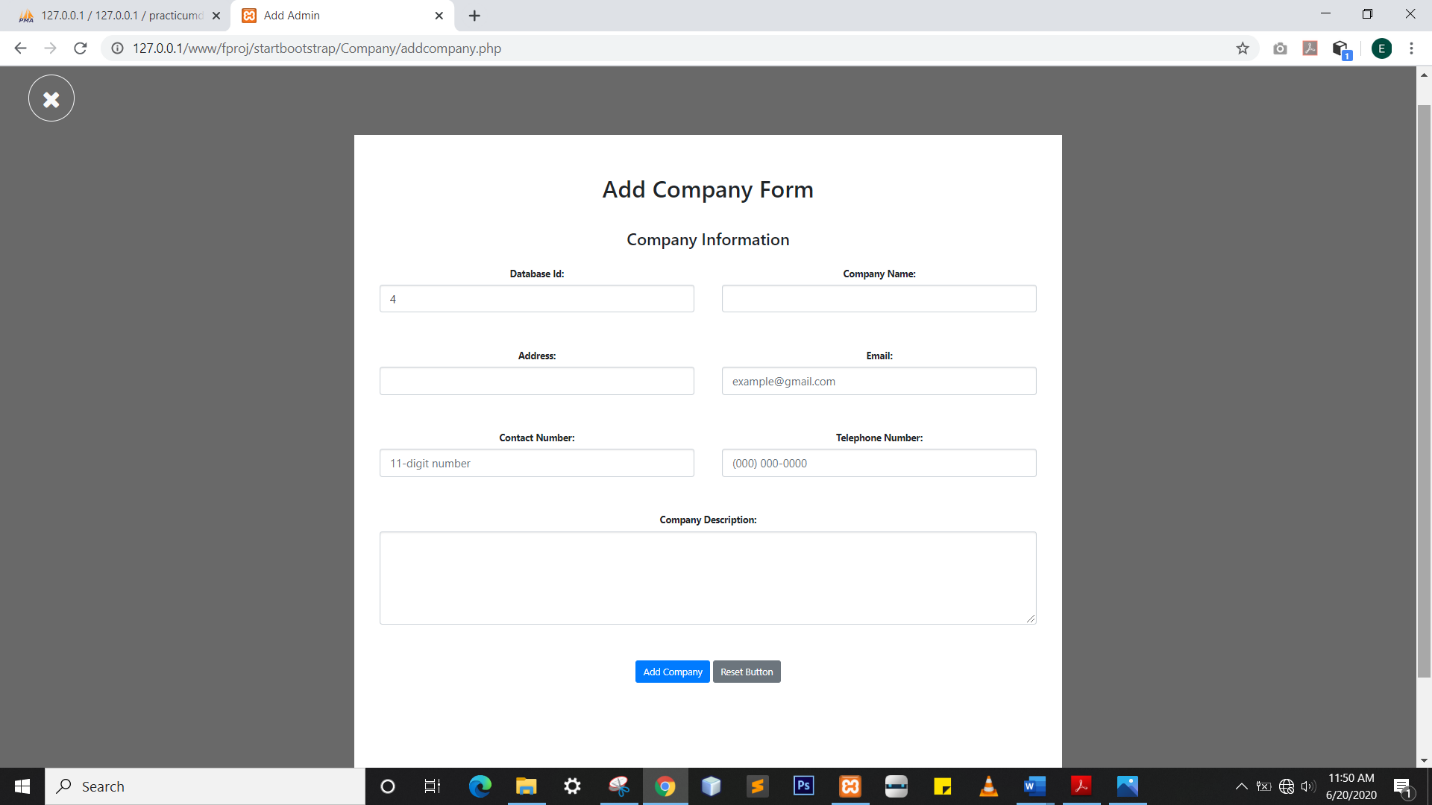
Enter Student records

Enter Instructor information

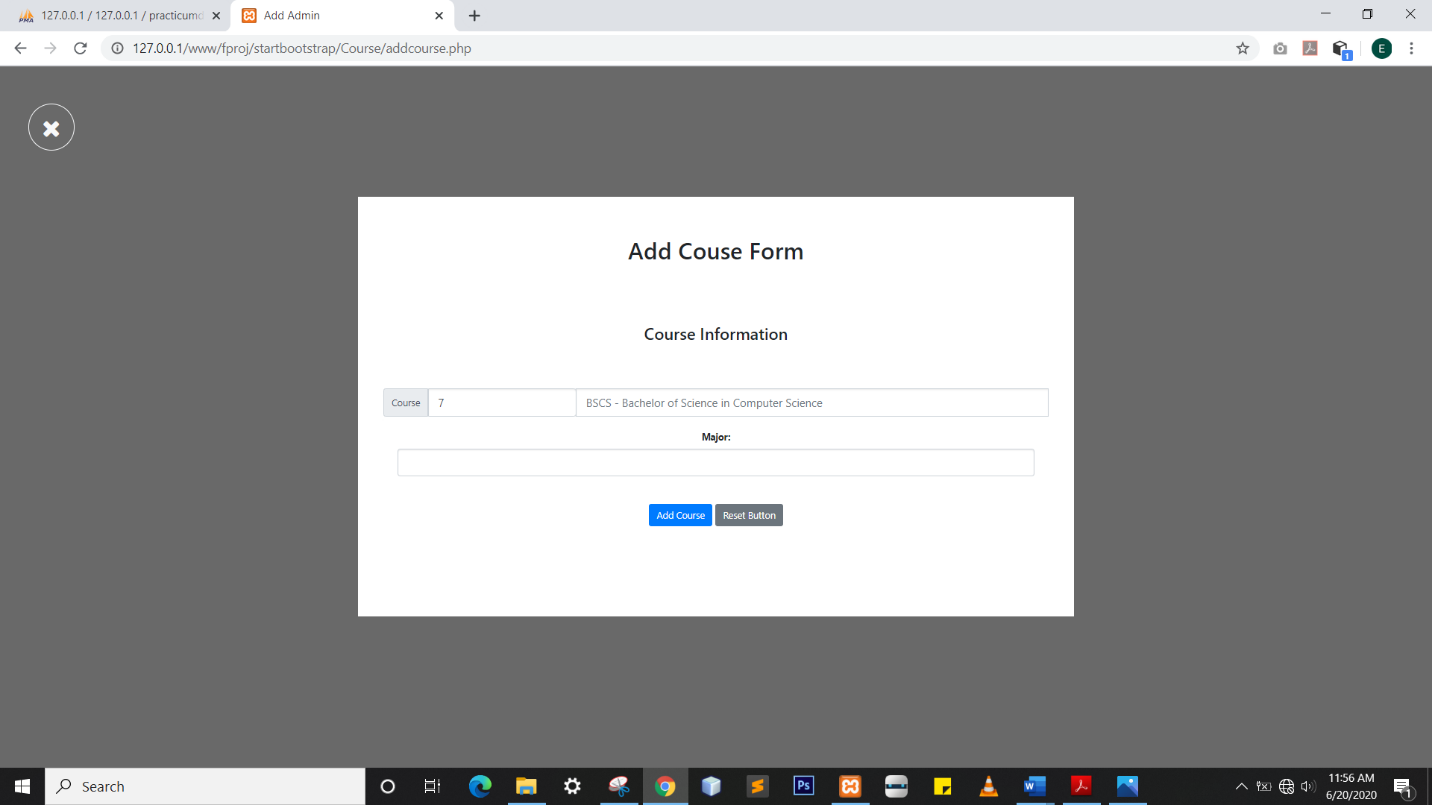
Enter admin information



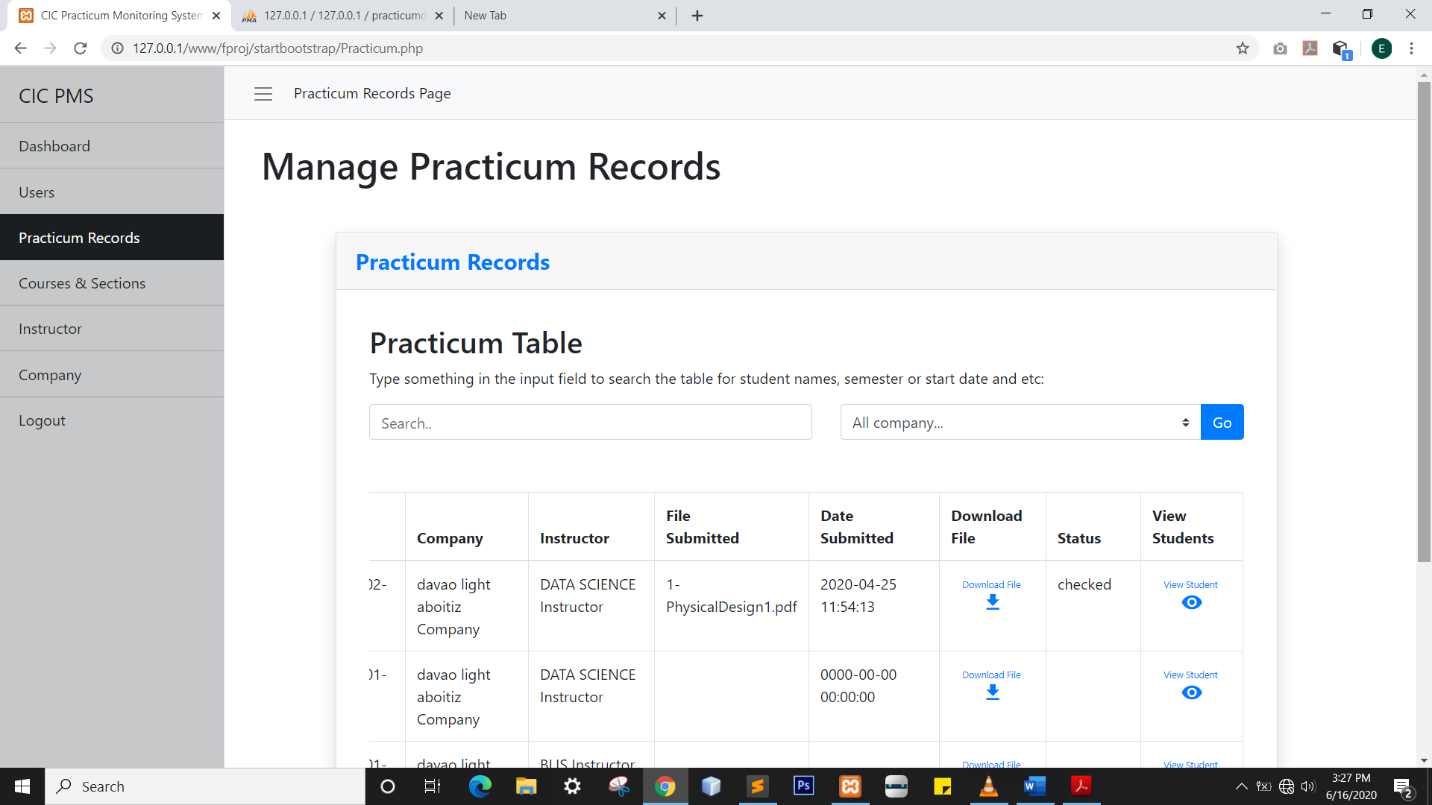
Enter company information



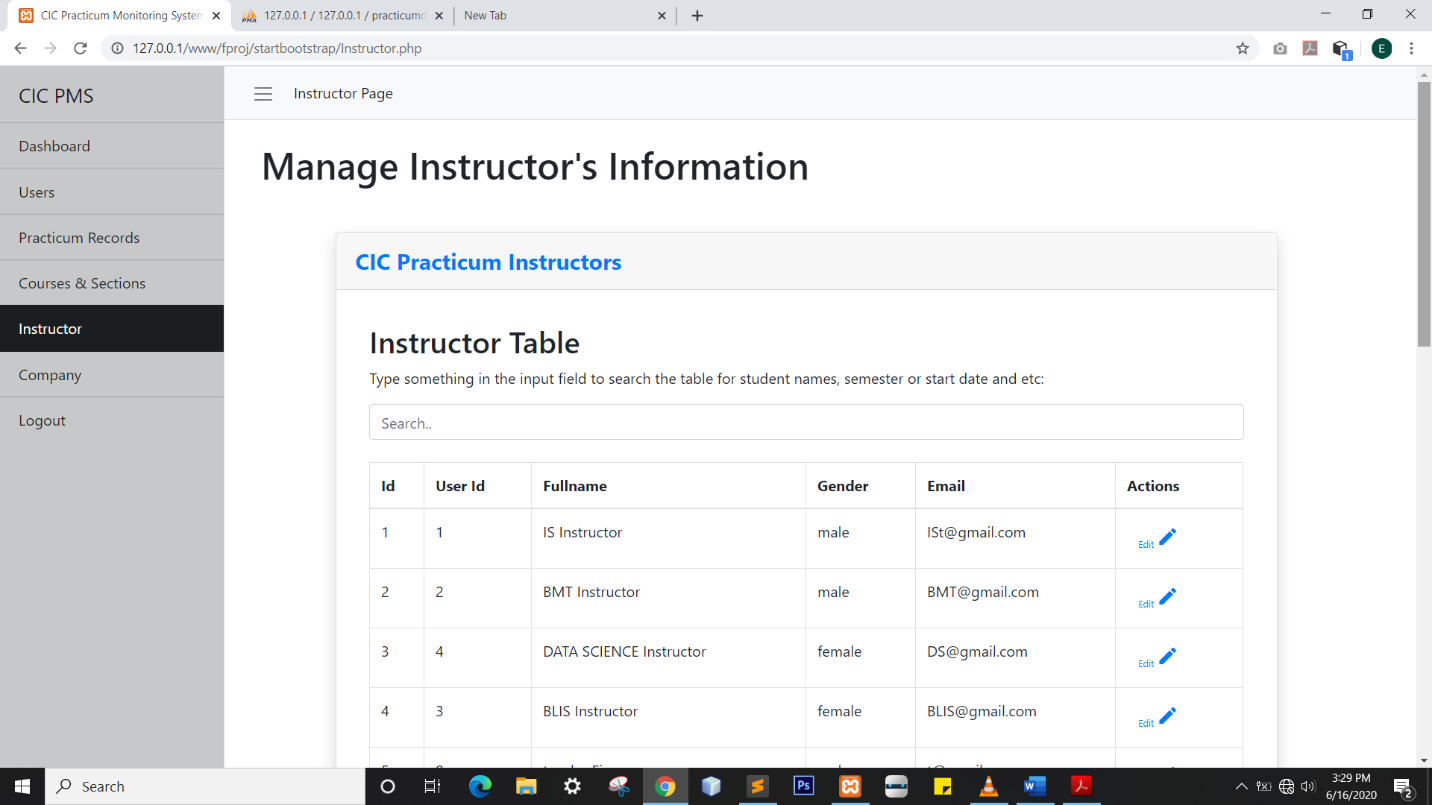
Enter section and course information



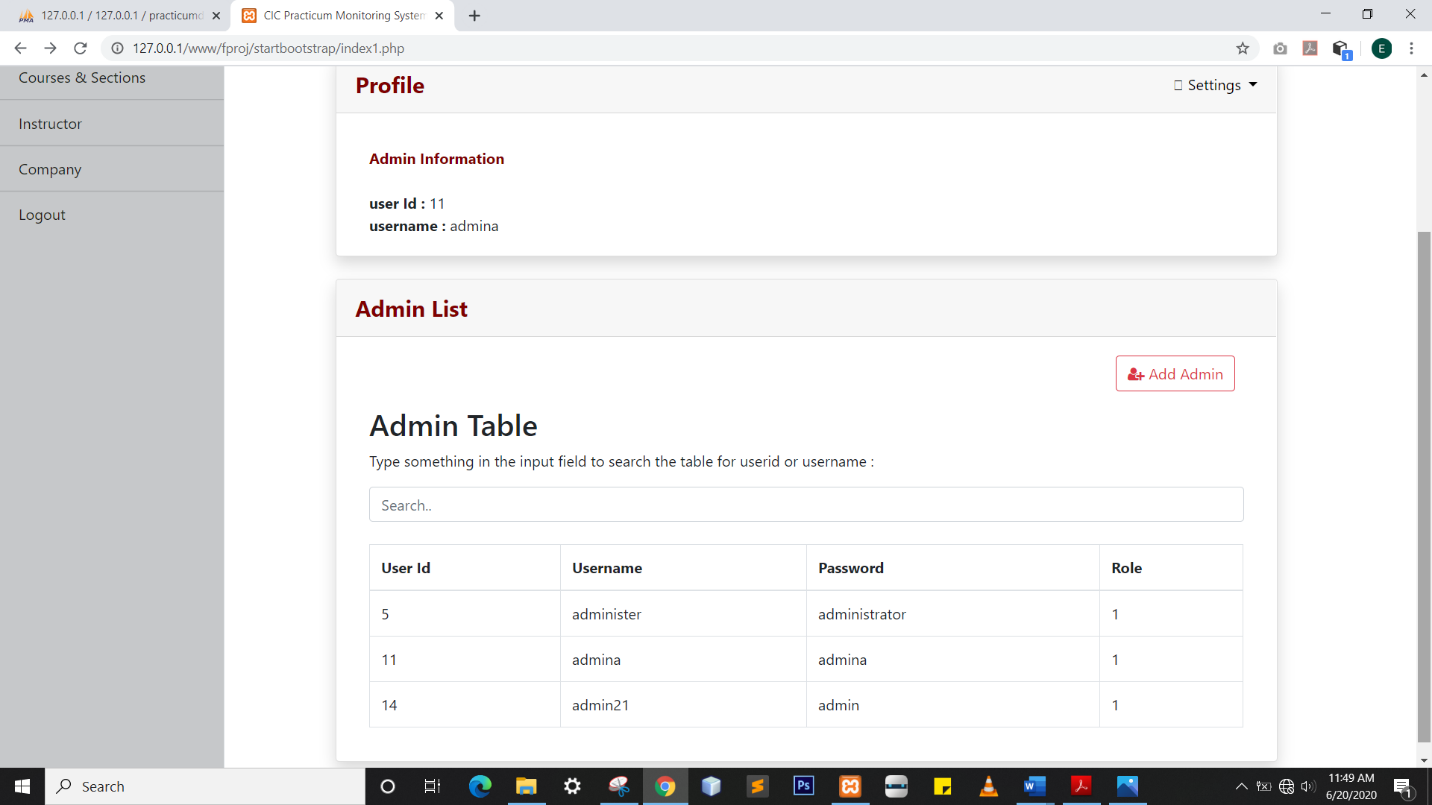
View Student records

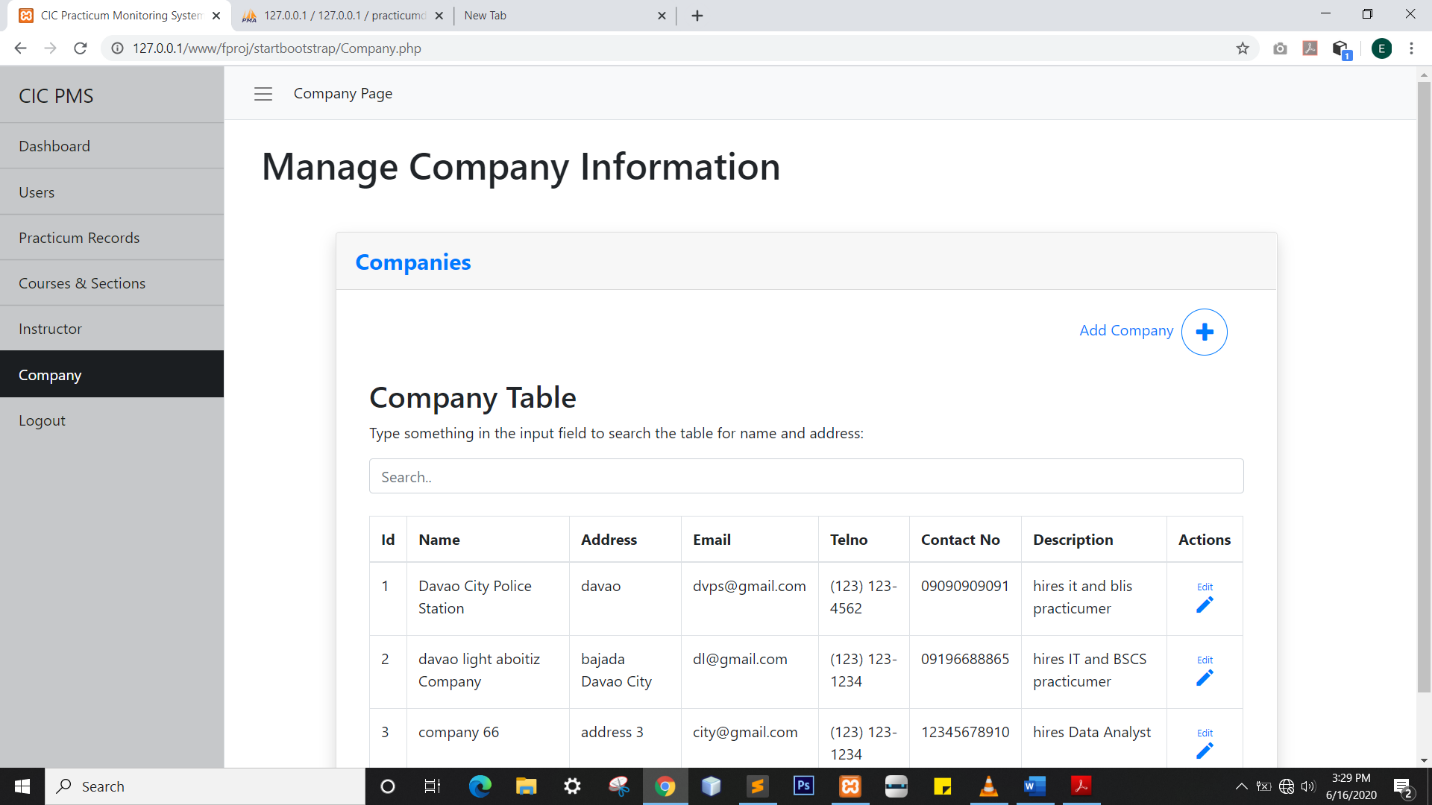


View Instructor information

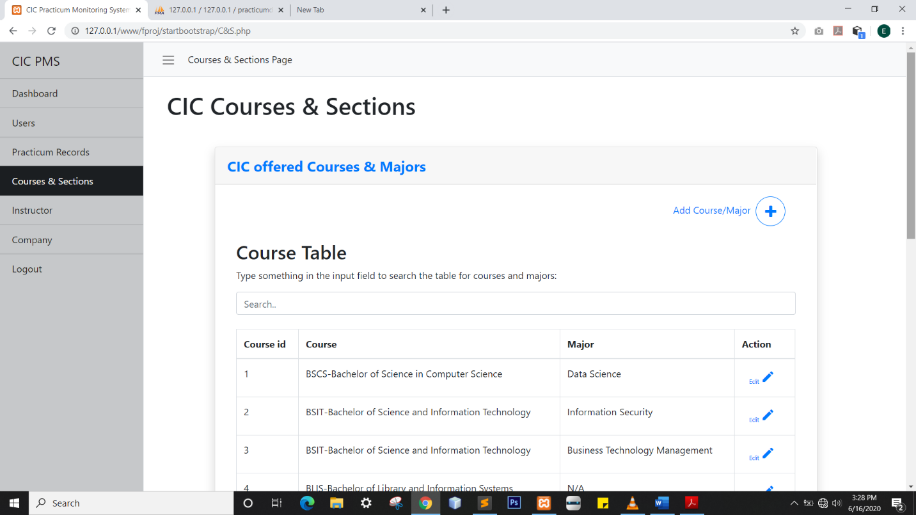


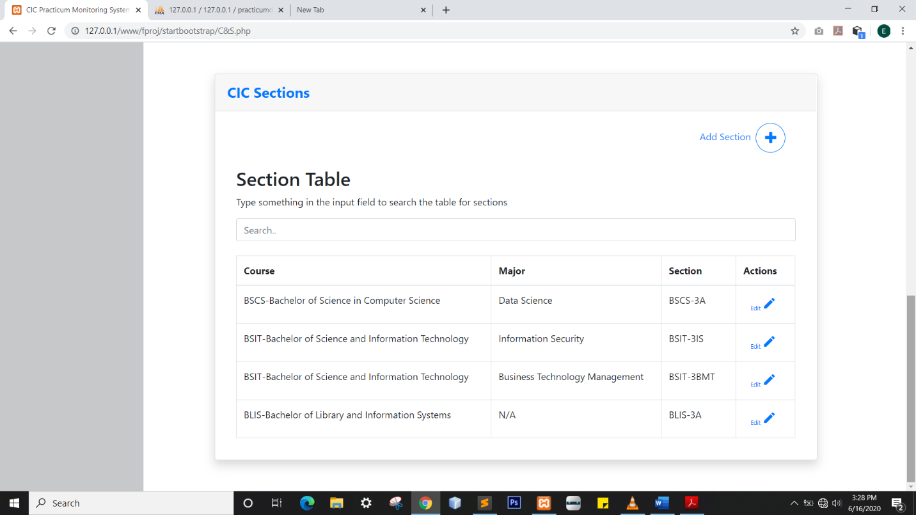
View admin information



View company information

View section and course information





**3.2 Conceptual Data Model**

*Entitiies*

User – Strong Emtity

Student – Weak Entity

Instructor – Weak Entity

Records – Weak Entity

Company -Strong Entity

*Relationship Cardinalities*

User is a Student one-to-one relationship

User is an Instructor one-to-one relationship

Instructor Views Student one-to-many relationship

Instructor Monitors Records one-to-many relationship

Student has Records one-to-one relationship

Company is part of Records one-to-many relationship

Attributes

User user\_id **{PK},** username, password, role, reset\_password, date\_modified

Student s\_id **{PK}**, fname, mname, lname, gender, Birthdate, email, contactno, section,course, major, instructor

Instructor i\_id **{PK},** fullname,gender**,** email

Records r\_id **{PK},** s\_id**,** semester**,** start\_date**,** end\_date**,** companyname**,** teacher**,** filename**,** sub\_date**,** report\_status

Companyc\_id **{PK},** c\_name**,** c\_address**,** c\_email**,** c\_telno**,** c\_contactno**,** c\_description

**Conceptual Data Model**

Is a

Student

User

s\_id **{PK}**

fname

mname

lname

gender

birthdate

email

contactno

section

course

major

instructor

user\_id **{PK}**

username

password

role

reset\_password

date\_modified

0..1

1..1

Is an

1..1

1..\*

Views

1..1

0..1

Instructor

i\_id **{PK}**

fullname

gender

email

Has

1..1

Records

r\_id **{PK}**

s\_id

semester

start\_date

end\_date

companyname

teacher

filename

sub\_date

report\_status

1..1

1..\*

Monitors

1..1

Is part of

1..\*

Company

c\_id **{PK}**

c\_name

c\_address

c\_email

c\_telno

c\_contactno

c\_description

1..1

**CHAPTER IV**

**LOGICAL DATABASE DESIGN**

**User** (user\_id  **,**username**,** password**,** role **,** reset\_password, date\_modified)

Primary Key: user\_id

**Student** (s\_id **,** fname**,** mname**,** lname**,** gender,birthdate**,** email**,** contactno**,** section**,** course**,** major**,** teacher,,i\_id, user\_id )

1:\*, the one side is Instructor therefore the primary key of Instructor becomes an attribute to entity Student

Primary Key: s\_id

Foreign Key :i\_id reference Instructor (i\_id)

1:1 this case has 2 option. a).unite the 2 entity .b)choose either of the two entity to be the strong entity.In this case I chose option b.) and choose User

1:1 this case has 2 option. a).unite the 2 entity .b)choose either of the two entity to be the strong entity.This time I chose option b.) and choose User

1:1 this case has 2 option. a).unite the 2 entity .b)choose either of the two entity to be the strong entity.This time I chose option b.) and choose User

Foreign Key: user\_id reference User (user\_id)

**Instructor** (i\_id **,** fullname**,** gender**,** email, ,user\_id)

Primary Key: i\_id

Foreign Key: user\_id reference User (user\_id)

**Records** (r\_id **{PK},** semester**,** start\_date**,** end\_date**,** companyname**,** instructor**,** filename**,** sub\_date**,** report\_status, i\_id **,** c\_id , s\_id )

Primary Kety: r\_id

1:\*, the one side is Instructor therefore the primary key of Instructor becomes an attribute to entity Records

Foreign Key: s\_id reference User (s\_id)

Foreign Key: i\_id reference Instructor (i\_id)

1:\*, the one side is Company therefore the primary key of Instructor becomes an attribute to entity Records

Foreign Key c\_id reference Company (c\_id),

**Company** (c\_id **{PK},** c\_name**,** c\_address**,** c\_email**,** c\_telno**,** c\_contactno**,** c\_description)

Primary Key: c\_id

**3.2 Logical l Data Model**

s\_id **{PK}**

i\_id **{FK}**

user\_id **{FK}**

fname

mname

lname

gender

birthdate

email

contactno

section

course

major

instructor

Is a

Student

User

user\_id **{PK}**

username

password

role

reset\_password

date\_modified

0..1

1..1

Is an

1..1

1..\*

i\_id **{PK}**

user\_id **{FK}**

fullname

gender

email

Views

1..1

0..1

Instructor

1..1

Has

1..\*

1..1

Monitors

1..1

r\_id **{PK}**

**s\_id {FK}**

c\_id **{FK}**

i\_id **{FK}**

semester

start\_date

end\_date

companyname

teacher

filename

sub\_date

report\_status

Records

Is part of

1..\*

Company

c\_id **{PK}**

c\_name

c\_address

c\_email

c\_telno

c\_contactno

c\_description

1..1

**Logical Database Model (Normalized)**

s\_id **{PK}**

user\_id {FK}

fname

mname

lname

gender

birthdate

i\_id {FK}

section\_id {FK}

email

contactno

0..1

1..1

user\_id **{PK}**

username

password

role

reset\_password

Is a

Student

User

Instructor

Is an

1..1

1..\*

Views

1..1

1..1

0..1

Has

i\_id **{PK}**

user\_id {FK}

fullname

gender

email

1..1

1..1

Monitors

Records

r\_id **{PK}**

s\_id {FK}

sem

start\_date

end\_date

c\_id {FK}

i\_id {FK}

filename

sub\_date

report\_status

1..\*

1..1

has

1..\*

InsSection

inssection\_id **{PK}**

i\_id{FK}

section\_id {FK}

SY

1..\*

Is part of

1..1

section\_id **{PK}**

section

course\_id {FK}

Section

1..\*

Is part of

1..\*

1..1

Is part of

c\_id **{PK}**

c\_name

c\_address

c\_email

c\_telno

c\_contactno

c\_description

Company

1..1

course\_id **{PK}**

course

major

Course

**Explaination:**

As shown above the number of tables has increase upon the observation of the absence of section for instructors. Knowing that section is multi-valued( instructors can handle many section ) , following the rules of normalization in first normal form which require all data in the cell must be atomic hence we fix the section and made another entity to satisfy the rule which is InSection . InsSection table will hold all the sections an instructor have handled . IN the midst of satisfying the rule a realization have taken place that Course must be included in the list of entities because in every section there is a corresponding course.

**CHAPTER V**

**PHYSICAL DATABASE DESIGN**

**Physical Data Model**

user\_id int **<<PK>>**

username varchar(30)

password varchar(20)

date\_modified datetime()

role int(1)

reset\_password varchar (50)

Student

s\_id varchar **(10)<<PK>>**

user\_id int <<FK>>

fname varchar (50)

mname varchar (50)

lname varchar (50)

gender varchar (1)

birthdate date

i\_id int <<FK>>

section\_id int <<FK>>

email varchar (100)

contactno varchar(11)

1..1

User

Is a

0..1

Is an

1..1

1..\*

Views

1..1

1..1

0..1

Instructor

Has

i\_id int **<<PK>>**

user\_id int <<FK>>

fullname varchar(100)

gender varchar(1)

email varchar(100)

1..1

1..1

Monitors

Records

r\_id int **<<PK>>**

s\_id int <<FK>>

sem varchar (15)

start\_date date

end\_date date

c\_id int <<FK>>

i\_id int <<FK>>

filename varchar(50)

sub\_date date

report\_status varchar(15)

1..\*

1..1

1..1

1..\*

has

InsSection

inssection\_id **{PK}**

i\_id{FK}

section\_id {FK}

SY int (10)

1..\*

Is part of

section\_id int **<<PK>>**

i\_id int <<FK>>

section varchar(10)

course\_id int <<FK>>

Section

1..\*

Is part of

c\_id int **<<PK>>**

c\_name varchar(50)

c\_address varchar(100)

c\_email varchar (50)

c\_telno varchar (15)

c\_contactno varchar (11)

c\_description varchar(500)

Company

1..1

1..\*

Is part of

course\_id int **<<PK>>**

course varchar(50)

major varchar(30)

Course

1..1

**Indexes Type: BTREE**

**Table 0.0 Indexes**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tables** | **User** | **Instructor** | **Student** | **Records** | **Section** | **Course** | **Company** |
| **Indexes** | user\_id | i\_id  user\_id | s\_id  i\_id  user\_id  section\_id | r\_id  c\_id  i\_id  s\_id | section\_id  i\_id  Course\_id | course\_id | c\_id |

**Estimated disk space requirements: Minimum10 GB**

**System Security: username and password**

**5.2 Final Data Dictionary**

# Company

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Description** | **Type** | **Null** | **Default** | **Domain** | **Links to** | **Key/Index** |
| c\_id | Company identifier | int(4) | No |  | Any positive integer |  | PK |
| c\_name |  | varchar(50) | No |  |  |  |  |
| c\_address |  | varchar(100) | No |  |  |  |  |
| c\_email |  | varchar(50) | No |  |  |  |  |
| c\_telno |  | varchar(15) | Yes | *NULL* | (000) 000-0000 |  |  |
| c\_contactno |  | varchar(11) | Yes | *NULL* |  |  |  |
| c\_description |  | varchar(500) | No |  |  |  |  |

# Course

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Description** | **Type** | **Null** | **Default** | **Domain** | **Links to** | **Key/Index** |
| course\_id | Course Identifier | int(2) | No |  | Any positive integer |  | PK |
| course |  | varchar(50) | No |  |  |  |  |
| major |  | varchar(30) | No |  |  |  |  |

# Instructor

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Description** | **Type** | **Null** | **Default** | **Domain** | **Links to** | **Key/Index** |
| i\_id | Instructor  Identifier | varchar(10) | No |  | Any positive integer |  | PK |
| user\_id |  | int(11) | No |  |  | user | FK |
| fullname |  | varchar(100) | No |  |  |  |  |
| gender |  | varchar(1) | No |  |  |  |  |
| email |  | varchar(100) | No |  |  |  |  |

# Records

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Description** | **Type** | **Null** | **Default** | **Domain** | **Links to** | **Key/Index** |
| r\_id | Record Identifier | int(10) | No |  | Any positive integer |  | PK |
| s\_id |  | int(10) | No |  |  | student | FK |
| sem |  | varchar(15) | Yes |  |  |  |  |
| start\_date |  | date | Yes |  |  |  |  |
| end\_date |  | date | Yes |  |  |  |  |
| c\_id |  | int(4) | Yes |  |  | company | FK |
| i\_id |  | varchar(10) | Yes |  |  | instructor | FK |
| filename |  | varchar(50) | Yes | *NULL* |  |  |  |
| sub\_date |  | datetime | Yes | *NULL* |  |  |  |
| report\_status |  | varchar(15) | Yes | *NULL* |  |  |  |

# Section

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Description** | **Type** | **Null** | **Default** | **Domain** | **Links to** | **Key/Index** |
| section\_id | Section Identifier | int(2) | No |  | Any positive integer |  | PK |
| section |  | varchar(10) | No |  |  |  |  |
| course\_id |  | int(2) | No |  |  | course | FK |

# InsSection

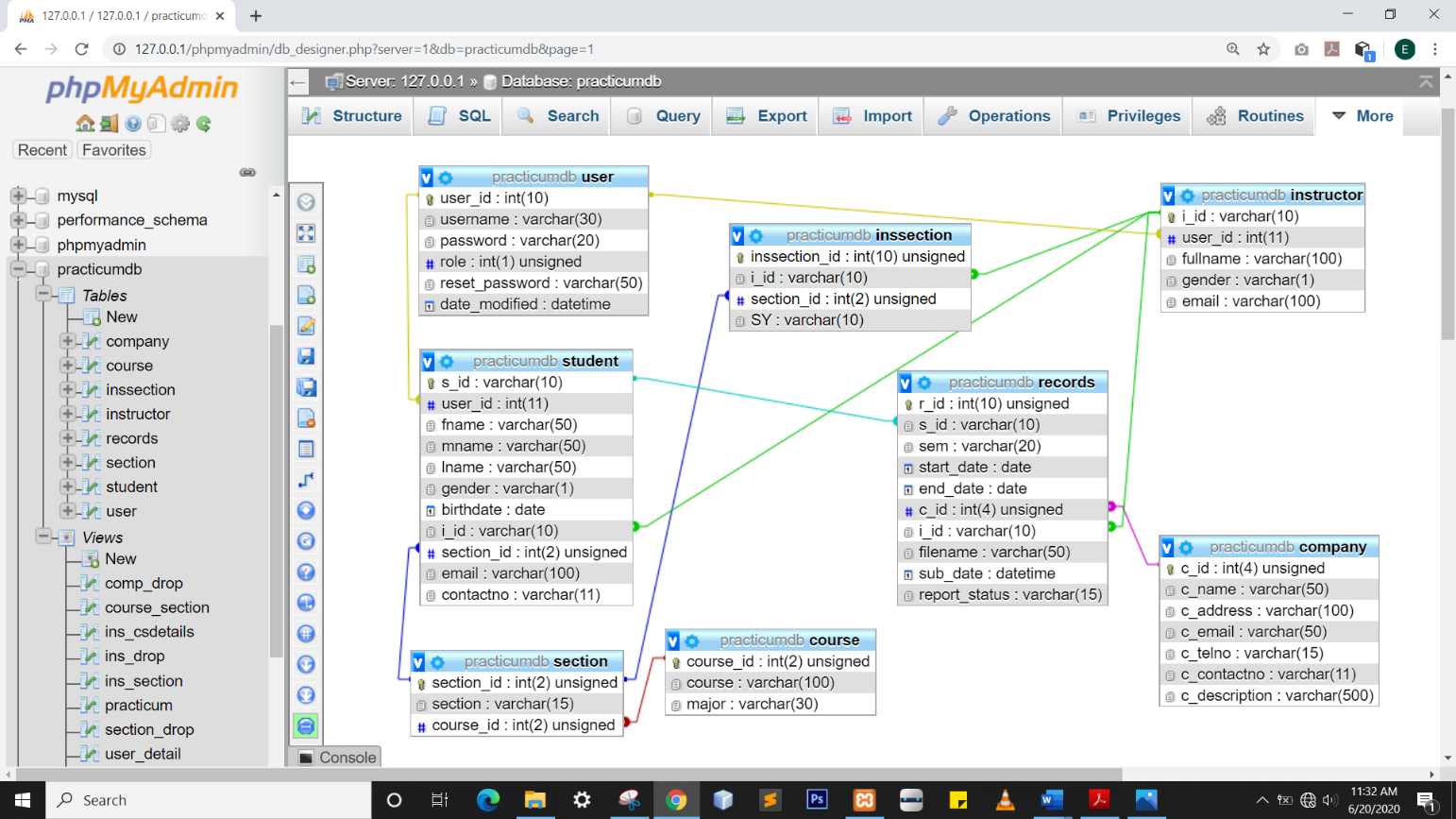
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Description** | **Type** | **Null** | **Default** | **Domain** | **Links to** | **Key/Index** |
| inssection\_id | Section Identifier | int(2) | No |  | Any positive integer |  | PK |
| i\_id |  | varchar(10) | No |  |  | instructor | FK |
| section\_id |  | int(2) | No |  |  | Section | FK |
| SY |  | varchar(10) | No |  |  |  |  |

# Student

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Description** | **Type** | **Null** | **Default** | **Domain** | **Links to** | **Key/Index** |
| s\_id | Student  Identifier | varchar(10) | No |  | 2000-12345 |  | PK |
| user\_id |  | int(11) | No |  |  | user | FK |
| fname |  | varchar(50) | No |  |  |  |  |
| mname |  | varchar(50) | No |  |  |  |  |
| lname |  | varchar(50) | No |  |  |  |  |
| gender |  | varchar(1) | No |  |  |  |  |
| birthdate |  | date | No |  |  |  |  |
| i\_id |  | varchar(10) | Yes |  |  | instructor | FK |
| section\_id |  | int(2) | No |  |  | section | FK |
| email |  | varchar(100) | No |  |  |  |  |
| contactno |  | varchar(11) | Yes | *NULL* |  |  |  |

# User

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Descrption** | **Type** | **Null** | **Default** | **Domain** | **Links to** | **Key/Index** |
| user\_id | User Identifier | int(10) | No |  | Any positive integer |  | PK |
| username |  | varchar(30) | No |  |  |  |  |
| password |  | varchar(20) | No |  |  |  |  |
| date\_modified |  | datetime() | Yes |  |  |  |  |
| role |  | int(1) | No |  |  |  |  |
| reset\_password |  | varchar(50) | No |  |  |  |  |

**5.3 Final Normalized Tables**

**5.4 Integrity Contraints**

**User & Student/Instructor/admin Cascade Constraints**

* Every user record must have corresponding Student/Instructor/admin record
* Every user record will not be allowed to be deleted
* Every user has only one account

**Student Record and Practicum Record**

* Every student record must have corresponding Practicum record
* Every record will not be allowed to be deleted

**Course & Section**

* Every course record must have corresponding section record
* Once a course record is updated the Course and Section will follow.
* Course and Section records are not allowed to be deleted.

**Section & Section History**

* Once a section record is updated the Section and Section History will follow

**Practicum Record & Company**

* Every practicum record must have a company .
* Once a company record is updated the Practicum and Company will follow

**5.5 Triggering Operations**

Back up company record

Insert

Company

If there is new company inserted

Insert new record

User Rule

Event

Condition

Entity Name

Action

Back up user record

Insert

user

If there is new user record inserted

Insert new record

Event

Event

Condition

Entity Name

Action

**5.6 Views**

Description

Course\_Section – shows every corresponding section of a course connecting course and section table

Code:

CREATE VIEW course\_section AS SELECT `practicumdb`.`course`.`course\_id` AS `course\_id`,

`practicumdb`.`course`.`course` AS `course`,

`practicumdb`.`course`.`major` AS `major`,

`practicumdb`.`section`.`section\_id` AS `section\_id`,

`practicumdb`.`section`.`section` AS `section`

FROM

(

`practicumdb`.`course`

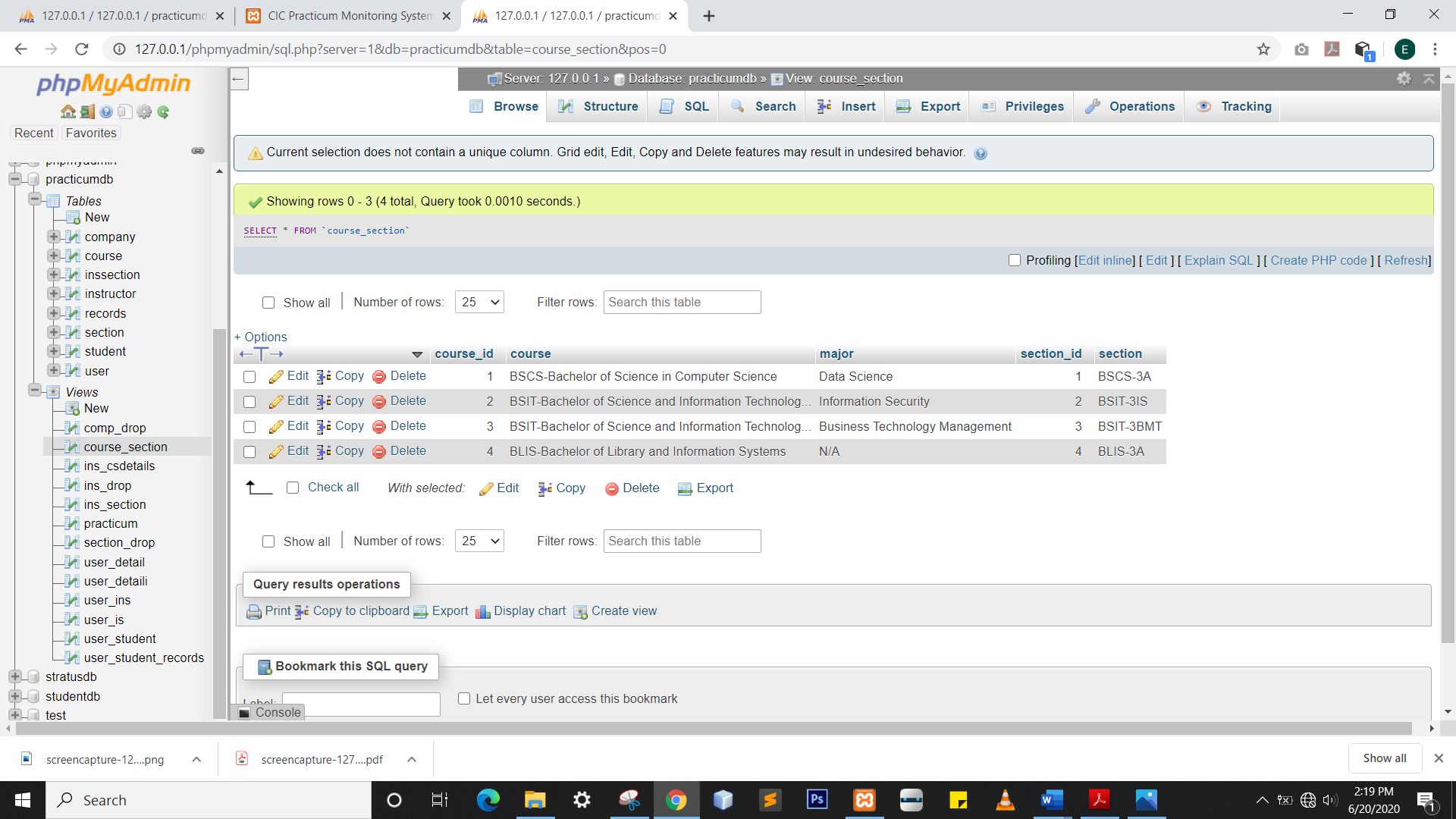
JOIN `practicumdb`.`section` ON

(

`practicumdb`.`course`.`course\_id` = `practicumdb`.`section`.`course\_id`

)

)



Screenshot

Description

ins\_csdetails – course\_section joins ins\_section to show corresponding course\_section to every instructor .

**5.7 Stored Procedures**

Code:

CREATE VIEW ins\_csdetails AS SELECT `ins\_section`.

`inssection\_id` AS `inssection\_id`,

`course\_section`.`course\_id` AS `course\_id`,

`course\_section`.`course` AS `course`,

`course\_section`.`major` AS `major`,

`ins\_section`.`section\_id` AS `section\_id`,

`ins\_section`.`section` AS `section`,

`ins\_section`.`user\_id` AS `user\_id`,

`ins\_section`.`i\_id` AS `i\_id`,

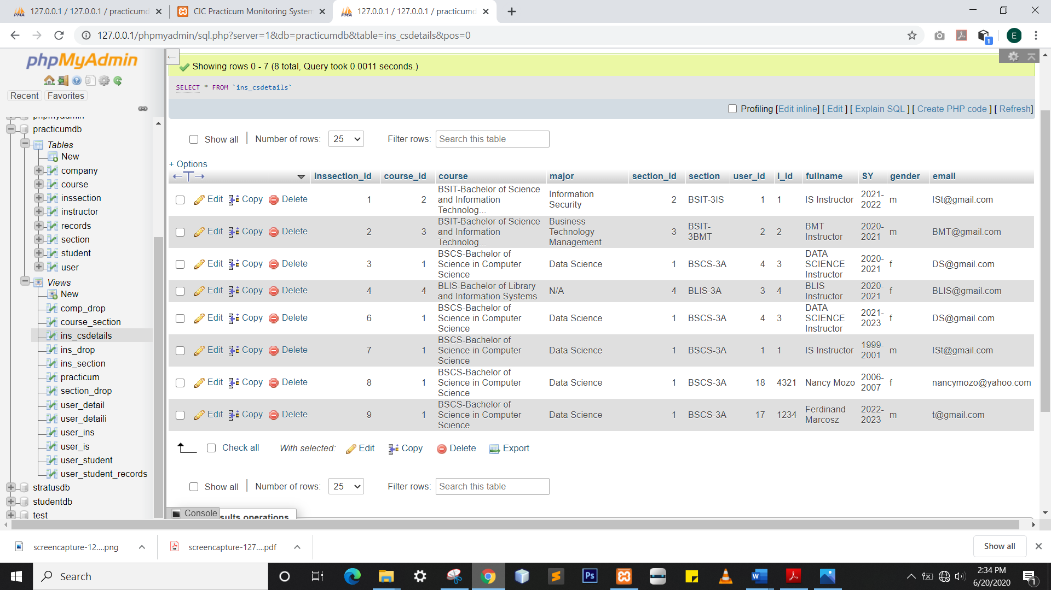
`ins\_section`.`fullname` AS `fullname`,

`ins\_section`.`SY` AS `SY`,`ins\_section`.

`gender` AS `gender`,

`ins\_section`.`email` AS `email` from (`practicumdb`.`course\_section` join `practicumdb`.`ins\_section` on(`course\_section`.`section\_id` = `ins\_section`.`section\_id`))

Screenshot

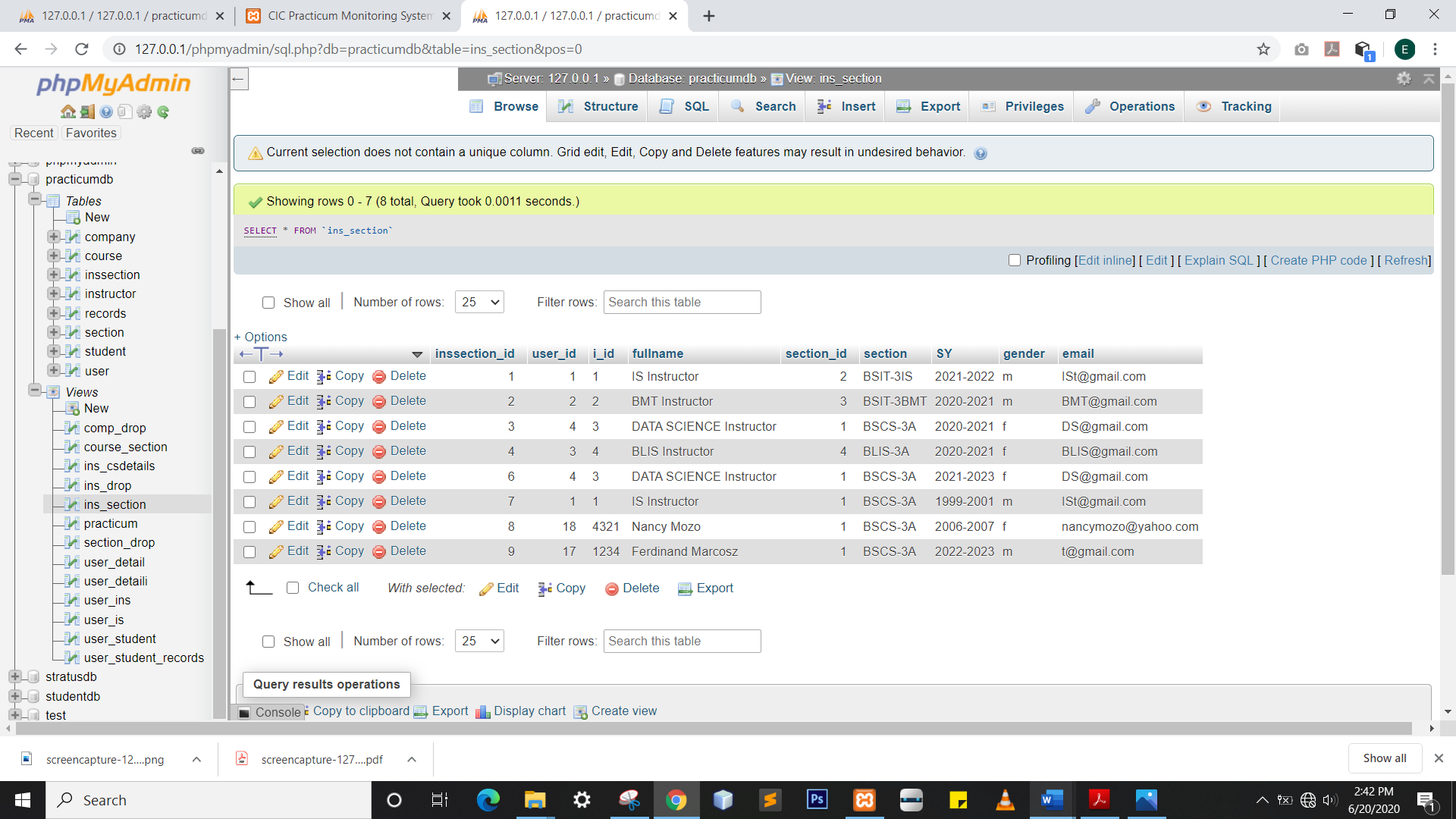


CREATE VIEW ins\_section AS SELECT `practicumdb`.`inssection`.`inssection\_id` AS `inssection\_id`,`practicumdb`.`instructor`.`user\_id` AS `user\_id`,`practicumdb`.`instructor`.`i\_id` AS `i\_id`,`practicumdb`.`instructor`.`fullname` AS `fullname`,`practicumdb`.`section`.`section\_id` AS `section\_id`,`practicumdb`.`section`.`section` AS `section`,`practicumdb`.`inssection`.`SY` AS `SY`,`practicumdb`.`instructor`.`gender` AS `gender`,`practicumdb`.`instructor`.`email` AS `email` from ((`practicumdb`.`inssection` join `practicumdb`.`instructor` on(`practicumdb`.`instructor`.`i\_id` = `practicumdb`.`inssection`.`i\_id`)) join `practicumdb`.`section` on(`practicumdb`.`inssection`.`section\_id` = `practicumdb`.`section`.`section\_id`))

Description

ins\_section - shows the corresponding section of an instructor.

Screenshot



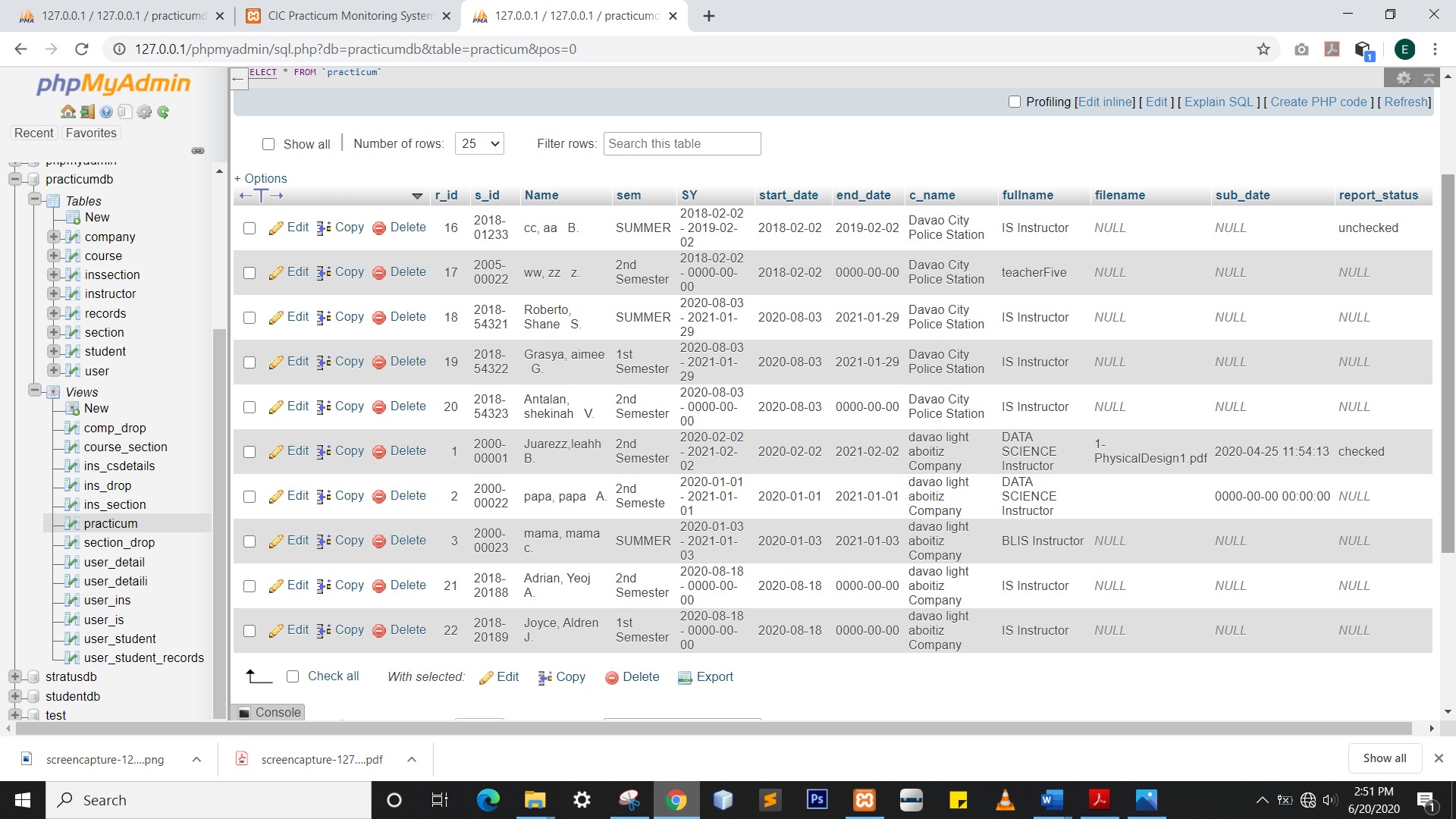
Description

Practicum- shows the corresponding information of every fk in records table.

Code

CREATE VIEW ins\_section AS SELECT `practicumdb`.`records`.`r\_id` AS `r\_id`,`practicumdb`.`records`.`s\_id` AS `s\_id`,concat(`practicumdb`.`student`.`lname`,',',`practicumdb`.`student`.`fname`,' ',`practicumdb`.`student`.`mname`) AS `Name`,`practicumdb`.`records`.`sem` AS `sem`,concat(`practicumdb`.`records`.`start\_date`,' - ',`practicumdb`.`records`.`end\_date`) AS `SY`,`practicumdb`.`records`.`start\_date` AS `start\_date`,`practicumdb`.`records`.`end\_date` AS `end\_date`,`practicumdb`.`company`.`c\_name` AS `c\_name`,`practicumdb`.`instructor`.`fullname` AS `fullname`,`practicumdb`.`records`.`filename` AS `filename`,`practicumdb`.`records`.`sub\_date` AS `sub\_date`,`practicumdb`.`records`.`report\_status` AS `report\_status` from (((`practicumdb`.`records` join `practicumdb`.`student` on(`practicumdb`.`student`.`s\_id` = `practicumdb`.`records`.`s\_id`)) join `practicumdb`.`company` on(`practicumdb`.`company`.`c\_id` = `practicumdb`.`records`.`c\_id`)) join `practicumdb`.`instructor` on(`practicumdb`.`instructor`.`i\_id` = `practicumdb`.`records`.`i\_id`))

Screenshot:

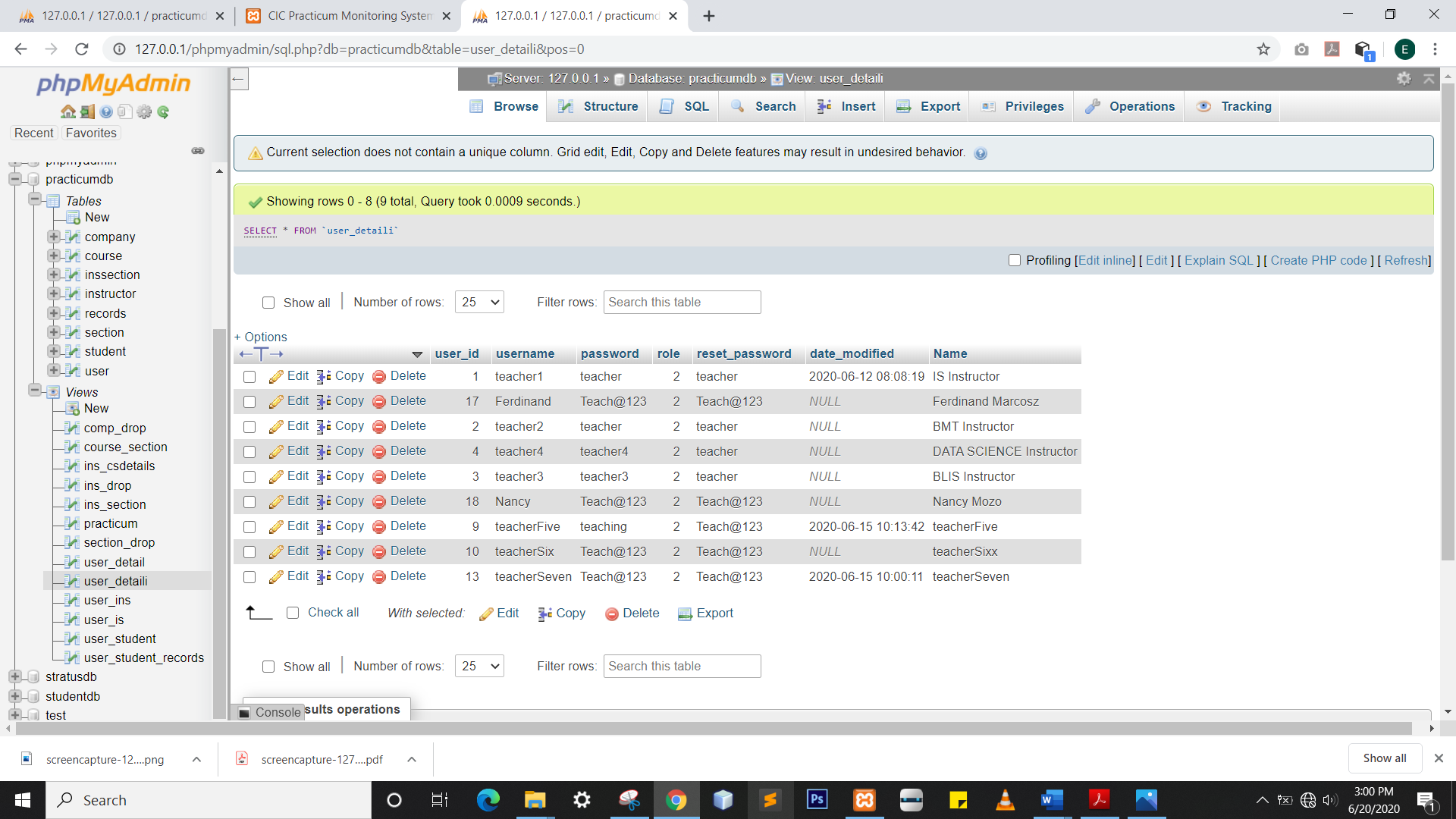


Description

user\_detaili shows the corresponding name in every row of user table from table instructor.

CREATE VIEW user\_detaili AS SELECT `practicumdb`.`user`.`user\_id` AS `user\_id`,`practicumdb`.`user`.`username` AS `username`,`practicumdb`.`user`.`password` AS `password`,`practicumdb`.`user`.`role` AS `role`,`practicumdb`.`user`.`reset\_password` AS `reset\_password`,`practicumdb`.`user`.`date\_modified` AS `date\_modified`,`practicumdb`.`instructor`.`fullname` AS `Name` from (`practicumdb`.`user` join `practicumdb`.`instructor` on(`practicumdb`.`user`.`user\_id` = `practicumdb`.`instructor`.`user\_id`))

Screenshot:

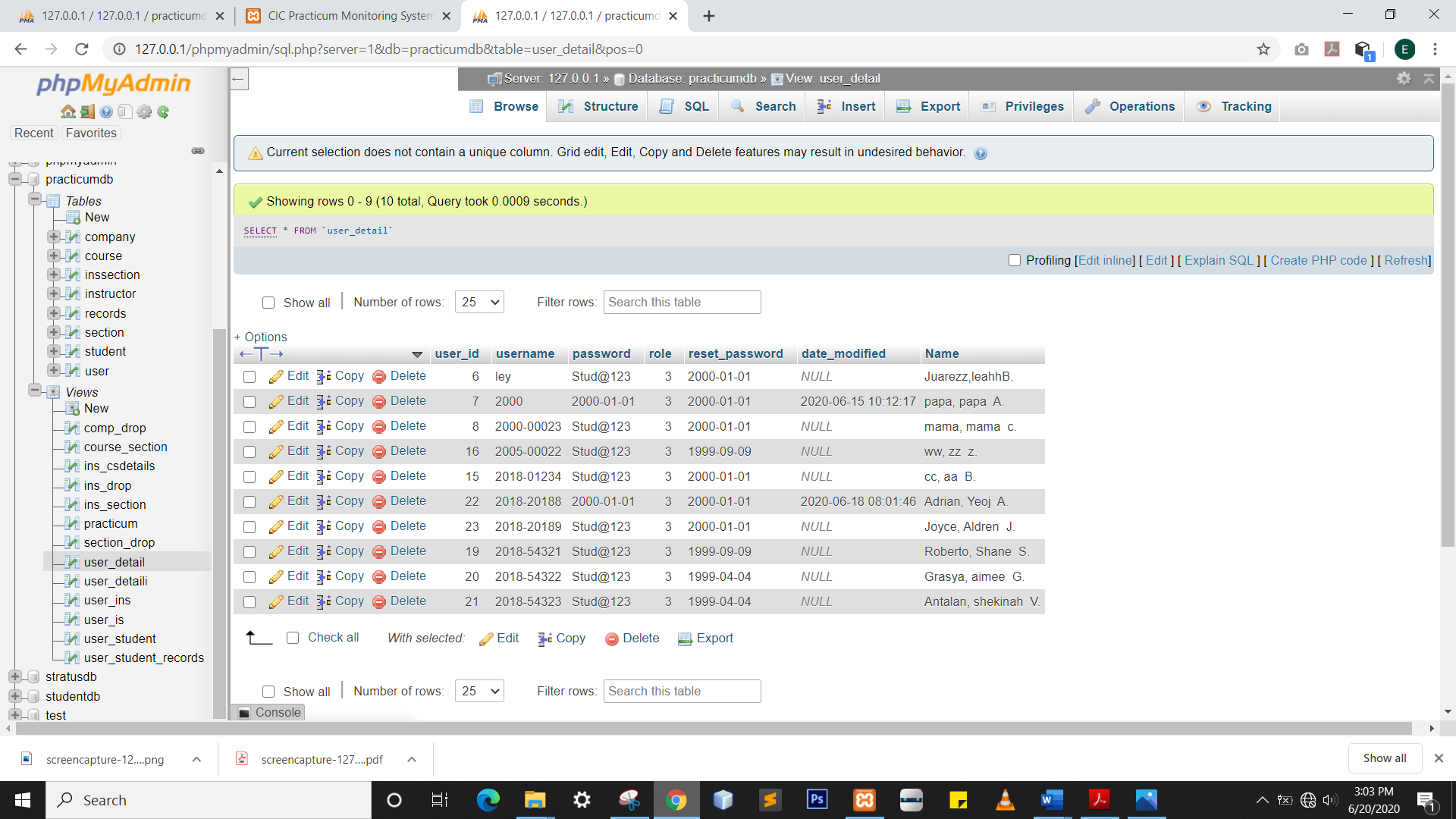


Description

user\_detail shows the corresponding name in every row of user table from table student.

CREATE VIEW user\_detail AS SELECT `practicumdb`.`user`.`user\_id` AS `user\_id`,`practicumdb`.`user`.`username` AS `username`,`practicumdb`.`user`.`password` AS `password`,`practicumdb`.`user`.`role` AS `role`,`practicumdb`.`user`.`reset\_password` AS `reset\_password`,`practicumdb`.`user`.`date\_modified` AS `date\_modified`,concat(`practicumdb`.`student`.`lname`,',',`practicumdb`.`student`.`fname`,`practicumdb`.`student`.`mname`) AS `Name` from (`practicumdb`.`user` join `practicumdb`.`student` on(`practicumdb`.`user`.`user\_id` = `practicumdb`.`student`.`user\_id`))

Screenshot :

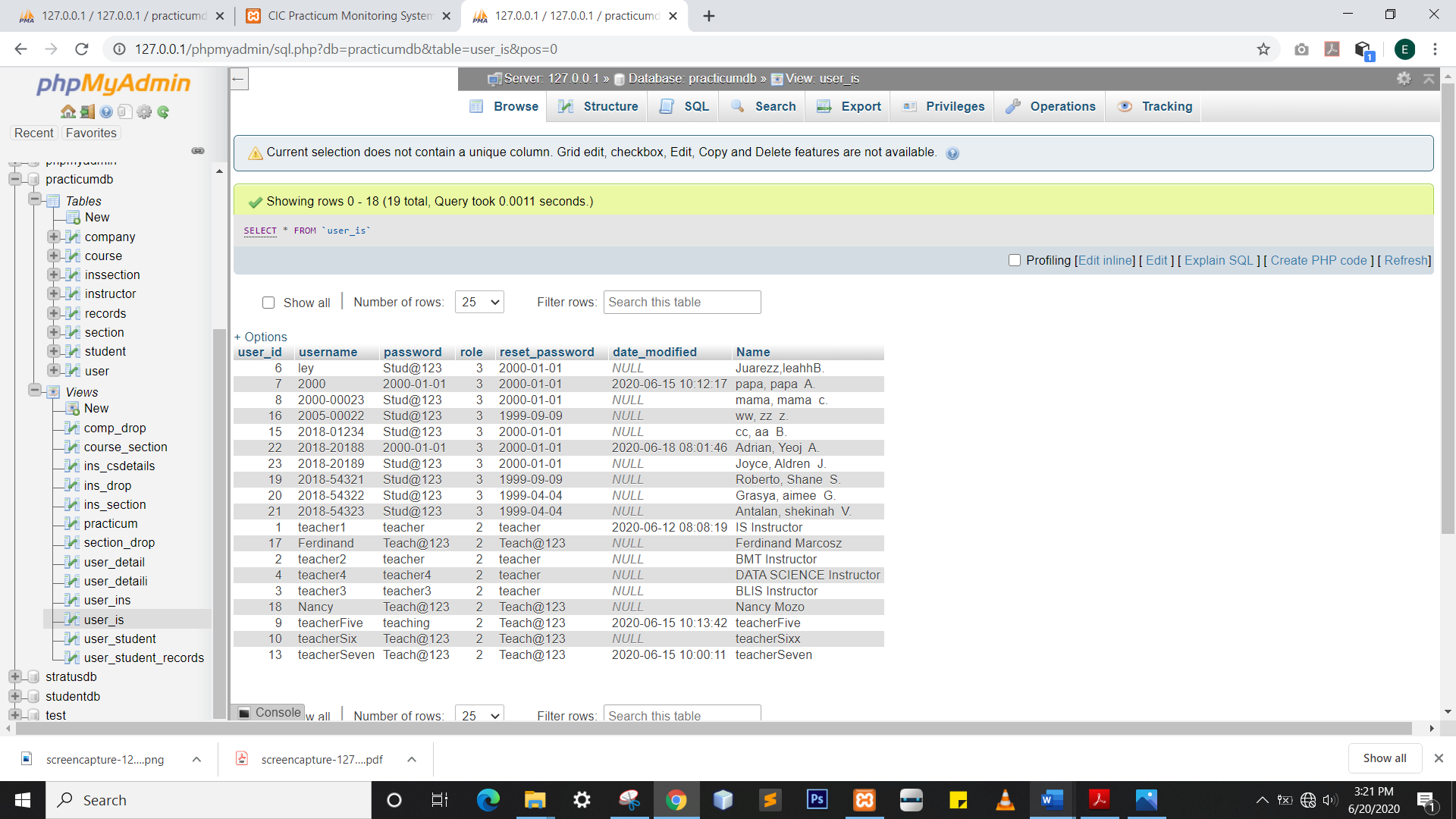


Description

user\_is shows the union of user\_detaili and user\_detail.

CREATE VIEW user\_detail AS SELECT `user\_detail`.`user\_id` AS `user\_id`,`user\_detail`.`username` AS `username`,`user\_detail`.`password` AS `password`,`user\_detail`.`role` AS `role`,`user\_detail`.`reset\_password` AS `reset\_password`,`user\_detail`.`date\_modified` AS `date\_modified`,`user\_detail`.`Name` AS `Name` from `practicumdb`.`user\_detail` union all select `user\_detaili`.`user\_id` AS `user\_id`,`user\_detaili`.`username` AS `username`,`user\_detaili`.`password` AS `password`,`user\_detaili`.`role` AS `role`,`user\_detaili`.`reset\_password` AS `reset\_password`,`user\_detaili`.`date\_modified` AS `date\_modified`,`user\_detaili`.`Name` AS `Name` from `practicumdb`.`user\_detaili`

Screeenshot:

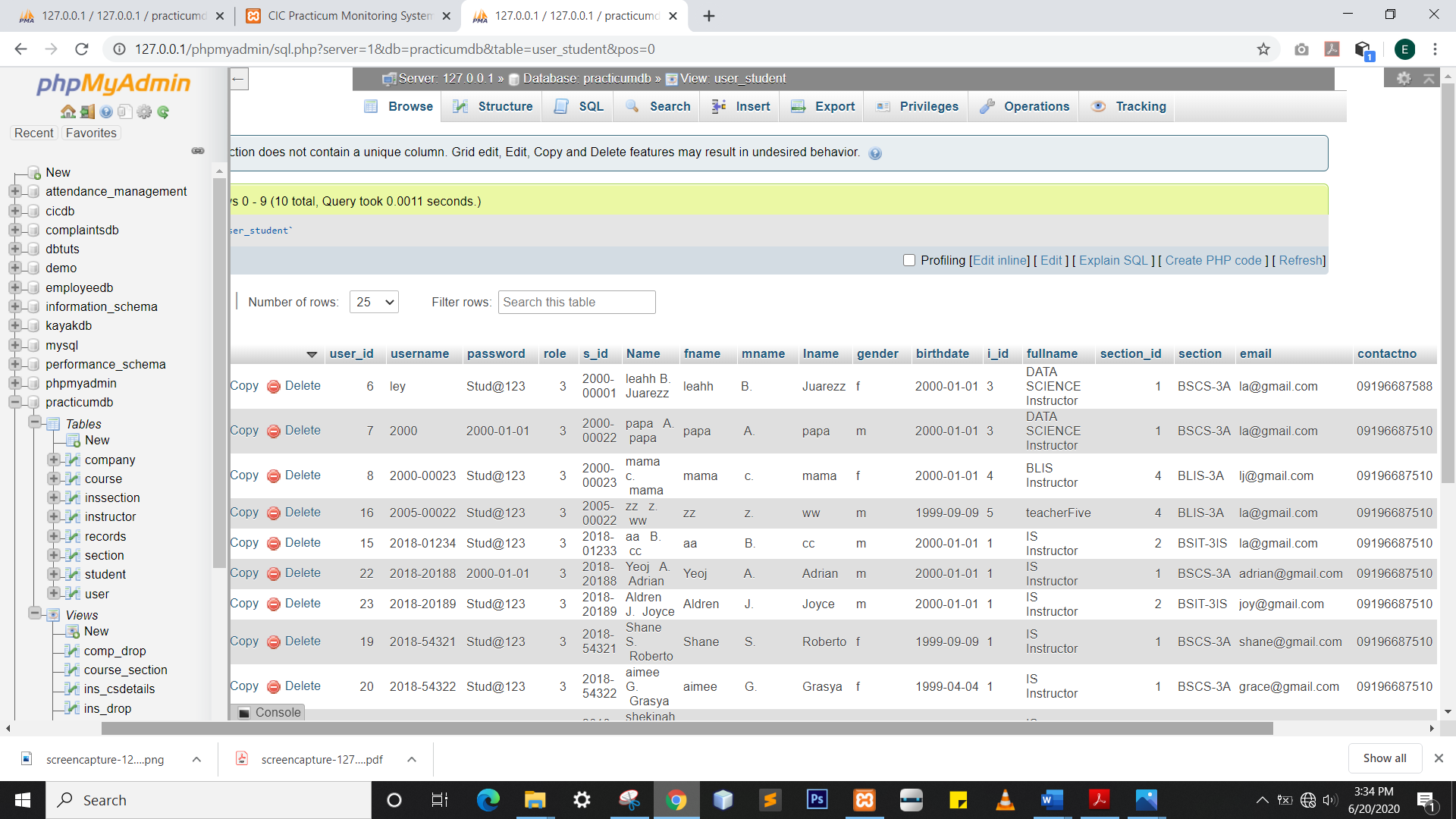


Description

User\_student joins table user and student.

CREATE VIEW user\_detail AS SELECT `practicumdb`.`user`.`user\_id` AS `user\_id`,`practicumdb`.`user`.`username` AS `username`,`practicumdb`.`user`.`password` AS `password`,`practicumdb`.`user`.`role` AS `role`,`practicumdb`.`student`.`s\_id` AS `s\_id`,concat(`practicumdb`.`student`.`fname`,' ',`practicumdb`.`student`.`mname`,' ',`practicumdb`.`student`.`lname`) AS `Name`,`practicumdb`.`student`.`fname` AS `fname`,`practicumdb`.`student`.`mname` AS `mname`,`practicumdb`.`student`.`lname` AS `lname`,`practicumdb`.`student`.`gender` AS `gender`,`practicumdb`.`student`.`birthdate` AS `birthdate`,`practicumdb`.`instructor`.`i\_id` AS `i\_id`,`practicumdb`.`instructor`.`fullname` AS `fullname`,`practicumdb`.`section`.`section\_id` AS `section\_id`,`practicumdb`.`section`.`section` AS `section`,`practicumdb`.`student`.`email` AS `email`,`practicumdb`.`student`.`contactno` AS `contactno` from (((`practicumdb`.`student` join `practicumdb`.`user` on(`practicumdb`.`student`.`user\_id` = `practicumdb`.`user`.`user\_id`)) join `practicumdb`.`instructor` on(`practicumdb`.`instructor`.`i\_id` = `practicumdb`.`student`.`i\_id`)) join `practicumdb`.`section` on(`practicumdb`.`section`.`section\_id` = `practicumdb`.`student`.`section\_id`))

Screenshot:

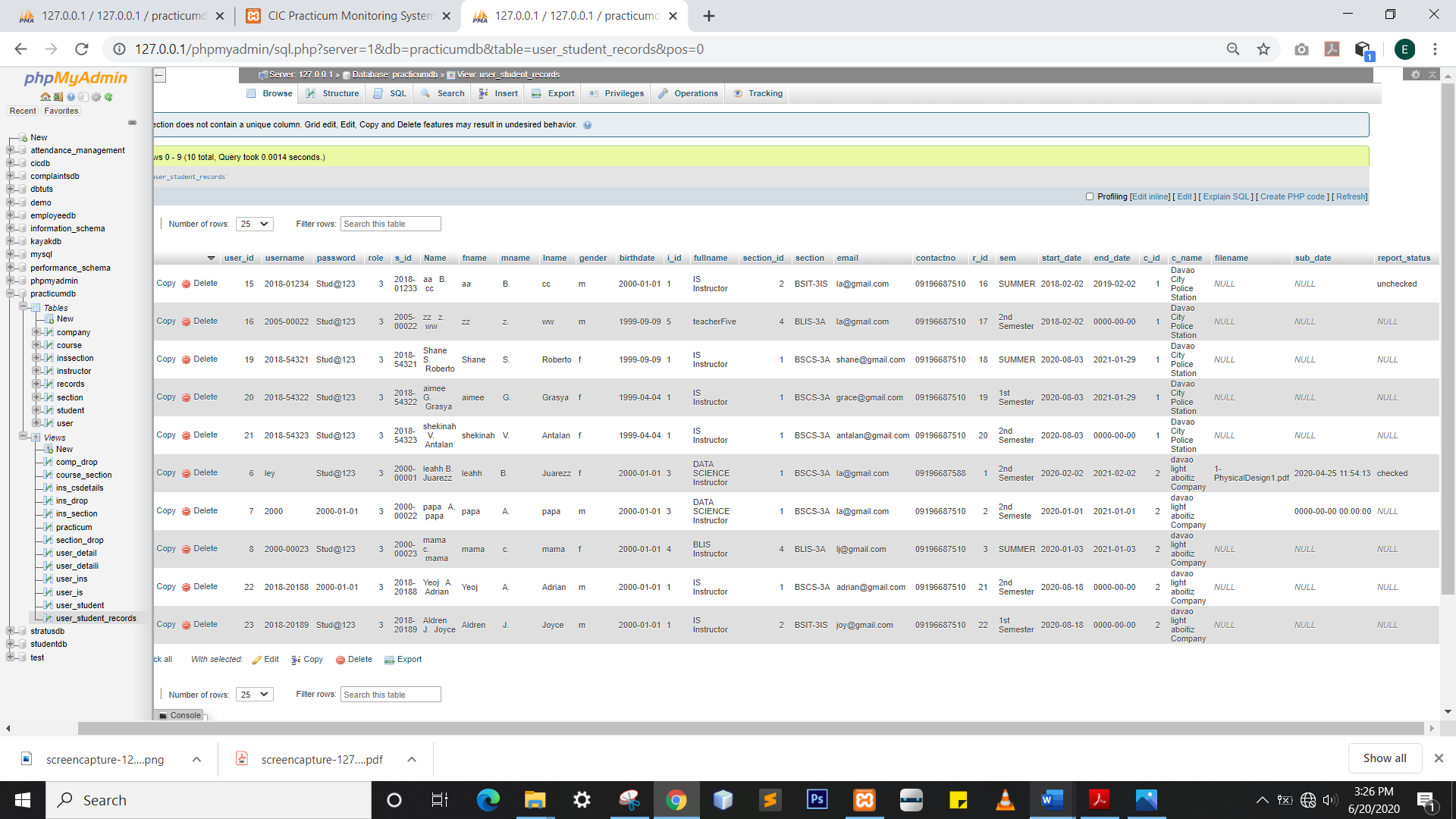


Description

User\_student\_records joins table user,student & records

CREATE VIEW user\_detail AS SELECT `user\_student`.`user\_id` AS `user\_id`,`user\_student`.`username` AS `username`,`user\_student`.`password` AS `password`,`user\_student`.`role` AS `role`,`user\_student`.`s\_id` AS `s\_id`,`user\_student`.`Name` AS `Name`,`user\_student`.`fname` AS `fname`,`user\_student`.`mname` AS `mname`,`user\_student`.`lname` AS `lname`,`user\_student`.`gender` AS `gender`,`user\_student`.`birthdate` AS `birthdate`,`user\_student`.`i\_id` AS `i\_id`,`user\_student`.`fullname` AS `fullname`,`user\_student`.`section\_id` AS `section\_id`,`user\_student`.`section` AS `section`,`user\_student`.`email` AS `email`,`user\_student`.`contactno` AS `contactno`,`practicumdb`.`records`.`r\_id` AS `r\_id`,`practicumdb`.`records`.`sem` AS `sem`,`practicumdb`.`records`.`start\_date` AS `start\_date`,`practicumdb`.`records`.`end\_date` AS `end\_date`,`practicumdb`.`company`.`c\_id` AS `c\_id`,`practicumdb`.`company`.`c\_name` AS `c\_name`,`practicumdb`.`records`.`filename` AS `filename`,`practicumdb`.`records`.`sub\_date` AS `sub\_date`,`practicumdb`.`records`.`report\_status` AS `report\_status` from ((`practicumdb`.`records` join `practicumdb`.`user\_student` on(`practicumdb`.`records`.`s\_id` = `user\_student`.`s\_id`)) join `practicumdb`.`company` on(`practicumdb`.`company`.`c\_id` = `practicumdb`.`records`.`c\_id`))

Screenshot:



**4.7 Stored Procedures and Functions**

Description

Stored procedure use to back up company records

Code:

DROP TRIGGER IF EXISTS `after\_company\_insert`;CREATE DEFINER=`root`@`localhost` TRIGGER `after\_company\_insert` AFTER INSERT ON `company` FOR EACH ROW INSERT INTO new\_company VALUES (NEW.c\_id,NEW.c\_name,NEW.c\_address,NEW.c\_email,NEW.c\_telno,NEW.c\_contactno,NEW.c\_description,NOW(),'insert')

DROP TRIGGER IF EXISTS `after\_user\_insert`;CREATE DEFINER=`root`@`localhost` TRIGGER `after\_user\_insert` AFTER INSERT ON `user` FOR EACH ROW INSERT INTO bu\_user VALUES (NEW.user\_id,NEW.username,NEW.password,NEW.role,NEW.reset\_password,NEW.date\_modified,NOW(),'insert')

Description

Stored procedure use to back up user records