**Open Ended Project Report**

**On**

Subject Enrolment System

Submitted for the partial fulfilment of Bachelor of Engineering

By

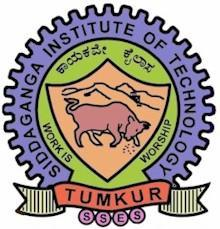
**Arup Das (1SI15CS015)**

**Under the guidance of**

Mr. C P [Prabodh](javascript:openMyPopUp%20('facultyprofile/15092017thejaswini%20S.pdf');)  M. Tech

Assistant Professor,

Department of CSE, SIT



**Department of Computer Science and Engineering**

**Siddaganga Institute of Technology, Tumkur – 572103**

(An Autonomous Institution, Affiliated to VTU, Belagavi & Recognized by AICTE, New Delhi)

**2017-2018**

***SYNOPSIS***

**PROJECT TITLE:** Subject Enrolment System

**PROBLEM STATEMENT:**

A college has many departments. Department would have students as well as faculty. The department belongs to many students and also employs many faculty members. A particular student and a faculty belongs to single department. A student enrols into many courses and a course can be studied by many students. Department offers many courses but a particular course belongs to a single department. A faculty teaches many courses but a particular course can be taught by many faculties.

The end users are the Administrative Office, Head of the departments, students and the faculties.

**SOLUTION:**

A database is created with the following entities and attributes:

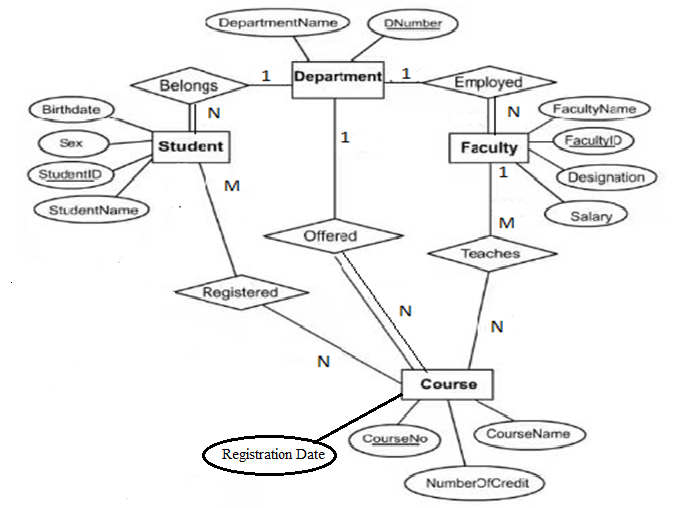
|  |  |
| --- | --- |
| **ENTITIES** | **ATTRIBUTES** |
| Student | StudentID, StudentName, Birthdate, Sex |
| Department | DNumber, DepartmentName |
| Faculty | FacultyID,FacultyName,Designation,Salary,DNumber |
| Course | CourseNo,CourseName,NumberOfCredits, Registration\_Date,Faculty\_ID,DNumber |

.

**RELATIONSHIPS:**

* Many students belong to one Department
* Department can employ many faculties.
* One Department can offer many courses.
* Many Faculties can teach a course.
* One Student can register many courses

**ER DIAGRAM:**



**RELATIONAL SCHEMA:**

**STUDENT**

|  |  |  |  |
| --- | --- | --- | --- |
| Birthdate | Sex | StudentID | StudentName |

**DEPARTMENT**

|  |  |  |
| --- | --- | --- |
| DepartmentName | DNumber | StudentID |

**FACULTY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FacultyName | FacultyID | Designation | Salary | DNumber |

**COURSE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CourseNo | CourseName | NumberOfCredits | Registration Date | FacultyID | DNumber |

**NORMALIZATION**

First Normal Form: This rule states that all the attributes in a relation must have atomic domains.

In the STUDENT table every student can have a single value for the attributes Birthdate, Sex, StudentID and StudentName. So it is 1NF.

Consider the DEPARTMENT table.

|  |  |  |
| --- | --- | --- |
| DepartmentName | DNumber | StudentID |
| CSE | 111 | 1SI15CS002, 1SI15CS013,1SI15CS015 |
| EC | 222 | 1SI16CS001, 1SI15CS011 |

Since many students can belong to DEPARTMENT so the table has multi-valued attributes. Hence we re-arrange the relation table as below, to convert it into 1NF.

|  |  |  |
| --- | --- | --- |
| DepartmentName | DNumber | StudentID |
| CSE | 111 | 1SI15CS002 |
| CSE | 111 | 1SI15CS013 |
| CSE | 111 | 1SI15CS015 |
| EC | 222 | 1SI16CS001 |
| EC | 222 | 1SI15CS011 |

Each attribute now contains a single value.

Every FACULTY has a unique Faculty\_ID, a fixed designation (a faculty can be an assistant professor or a professor or a HOD) and salary. Also a faculty can belong to a single department. Hence the table FACULTY is in 1NF.

Consider the COURSE table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CourseNo | CourseName | NumberOfCredits | Registration Date | FacultyID |
| Csiv01 | Programming | 4 | 02.01.2018 | Cs01,Cs02 |
| Csia02 | Web | 3 | 02.012016 | Cs03,Cs04 |

Since many faculties can teach the same course. Hence the table contains a multi-valued attribute- Faculty\_ID. So we re-arrange the relation table as below, to convert it into 1NF.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CourseNo | CourseName | NumberOfCredits | Registration Date | FacultyID |
| Csiv01 | Programming | 4 | 02.01.2018 | Cs01 |
| Csiv01 | Programming | 4 | 02.01.2018 | Cs02 |
| Csia02 | Web | 3 | 02.012016 | Cs03 |
| Csia02 | Web | 3 | 02.012016 | Cs04 |

Each attribute now contains an atomic value.

Second Normal Form: This rule states that every non-prime attribute should be fully functionally dependent on prime key attribute in a relation.

Consider the STUDENT table.

StudentID in Student table can be used to uniquely identify the rows in the STUDENT table.

|  |  |  |  |
| --- | --- | --- | --- |
| Birthdate | Sex | StudentID | StudentName |

Consider the DEPARTMENT table.

|  |  |  |
| --- | --- | --- |
| DepartmentName | DNumber | StudentID |

Here the primary key attributes are DNumber and StudentID. But both DNumber and StudentID can individually and uniquely identify the entries of the attribute DepartmentName. So the above table is not in 2NF.

Hence we break the relation into two as shown below. Now there is no partial dependency.

|  |  |
| --- | --- |
| DepartmentName | DNumber |

|  |  |
| --- | --- |
| DepartmentName | StudentID |

Consider the FACULTY table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FacultyName | FacultyID | Designation | Salary | DNumber |

As all the non-prime key attributes are totally dependent on the primary key attribute FacultyID. The above table is in 2NF.

Consider the COURSE table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CourseNo | CourseName | NumberOfCredits | Registration Date | FacultyID | DNumber |

Here the primary key attributes are CourseNo and FacultyID. But the attributes CourseName, NumberOfCredits and RegistrationDate are partially dependent on CourseNo and DNumber is partially dependent on FacultyID. Therefore, the table is not in 2NF.

Hence we break the relation into two as shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| CourseNo | CourseName | NumberOfCredits | Registration Date |

|  |  |
| --- | --- |
| FacultyID | DNumber |

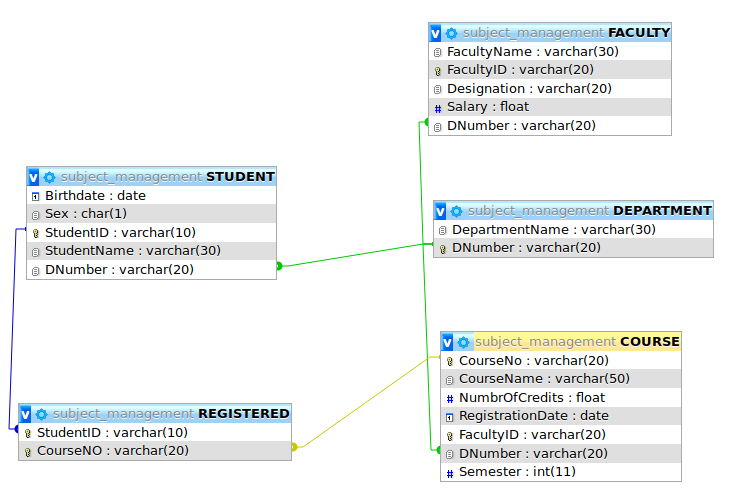
Now there is no partial dependency.

Third Normal Form: A table is in Third Normal Form if it satisfies the following conditions:

* It is in Second Normal Form.
* There is no transitive functional dependency.

As all the tables are in 2NF and there is no transitive functional dependencies in the tables. Hence all the tables are in 3NF.

**Design Structure**

****

**Creating table DEPARTMENT**

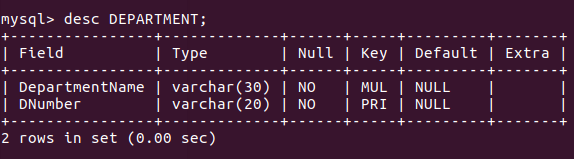
CREATE TABLE `DEPARTMENT` (

`DepartmentName` varchar(30) NOT NULL,

`DNumber` varchar(20) NOT NULL,

PRIMARY KEY (`DNumber`)

)



--Inserting data for table ` DEPARTMENT `--

INSERT INTO `DEPARTMENT` (`DepartmentName`, `DNumber`) VALUES

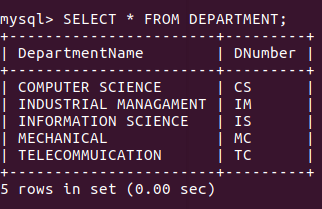
('COMPUTER SCIENCE', 'CS'),

('INDUSTRIAL MANAGAMENT', 'IM'),

('INFORMATION SCIENCE', 'IS'),

('MECHANICAL', 'MC'),

('TELECOMMUICATION', 'TC');



**Creating table STUDENT**

CREATE TABLE `STUDENT` (

`Birthdate` date NOT NULL,

`Sex` char(1) NOT NULL,

`StudentID` varchar(10) NOT NULL,

`StudentName` varchar(30) NOT NULL,

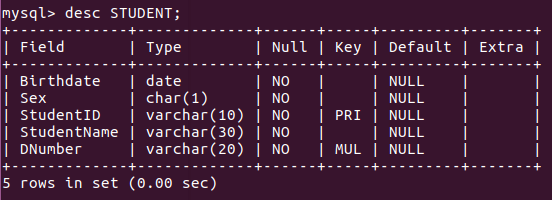
`DNumber` varchar(20) NOT NULL,

PRIMARY KEY (`StudentID`),

CONSTRAINT `FK\_DNumber`

FOREIGN KEY (`DNumber`) REFERENCES `DEPARTMENT`(`DNumber`);

)



--Inserting data for table ` STUDENT `--

INSERT INTO `STUDENT` (`Birthdate`, `Sex`, `StudentID`, `StudentName`, `DNumber`) VALUES

('1997-06-16', 'M', '1SI15CS002', 'Abhishek Koul', 'CS'),

('1997-09-12', 'M', '1SI15CS015', 'Arup Das', 'CS'),

('1996-07-24', 'M', '1SI15CS036', 'Divyanshu Anand', 'CS'),

('1997-03-26', 'M', '1SI15IM02', 'Abhishek Gowda', 'IM'),

('1997-03-03', 'M', '1SI15IM053', 'Privthi Raj', 'IM'),

('1997-09-11', 'M', '1SI15MC013', 'Antariksh Gupta', 'CS'),

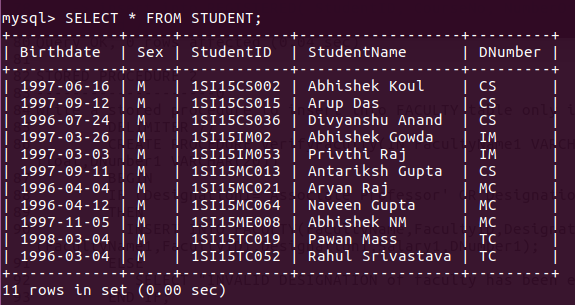
('1996-04-04', 'M', '1SI15MC021', 'Aryan Raj', 'MC'),

('1996-04-12', 'M', '1SI15MC064', 'Naveen Gupta', 'MC'),

('1997-11-05', 'M', '1SI15ME008', 'Abhishek NM', 'MC'),

('1998-03-04', 'M', '1SI15TC019', 'Pawan Mass', 'TC'),

('1996-03-04', 'M', '1SI15TC052', 'Rahul Srivastava', 'TC');



**Creating Table FACULTY**

CREATE TABLE `FACULTY` (

`FacultyName` varchar(30) NOT NULL,

`FacultyID` varchar(20) PRIMARY KEY,

`Designation` varchar(20) NOT NULL,

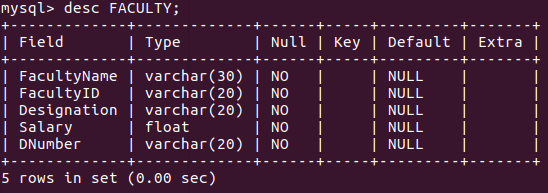
`Salary` float NOT NULL,

`DNumber` varchar(20) NOT NULL,

CONSTRAINT `FK\_FACULTY\_

FOREIGN KEY(`DNumber`) REFERENCES FACULTY (`DNumber`);

)



--Inserting data for table ` FACULTY `--

INSERT INTO `FACULTY` (`FacultyName`, `FacultyID`, `Designation`, `Salary`, `DNumber`) VALUES

('Ram Pravesh', 'CS01', 'Professor', 100000, 'CS'),

('Ashwini N.S.', 'CS02', 'Assistant Professor', 65000, 'CS'),

('Ashwini B.P.', 'CS03', 'Assistant Professor', 65000, 'CS'),

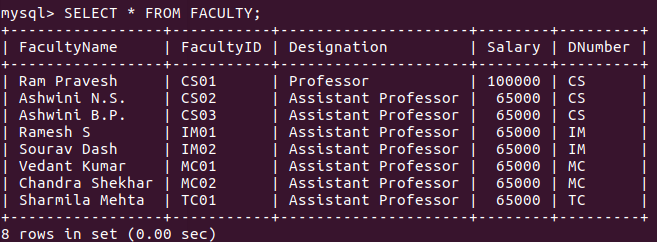
('Ramesh S', 'IM01', 'Assistant Professor', 65000, 'IM'),

('Sourav Dash', 'IM02', 'Assistant Professor', 65000, 'IM'),

('Vedant Kumar', 'MC01', 'Assistant Professor', 65000, 'MC'),

('Chandra Shekhar', 'MC02', 'Assistant Professor', 65000, 'MC'),

('Sharmila Mehta', 'TC01', 'Assistant Professor', 65000, 'TC');



**Creating Table COURSE**

CREATE TABLE `COURSE` (

`CourseNo` varchar(20) NOT NULL,

`CourseName` varchar(50) NOT NULL,

`NumbrOfCredits` float NOT NULL,

`RegistrationDate` date DEFAULT NULL,

`FacultyID` varchar(20) NOT NULL,

`DNumber` varchar(20) DEFAULT NULL,

`Semester` int(11) DEFAULT NULL.

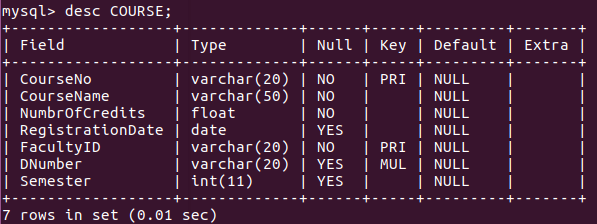
PRIMARY KEY (`CourseNo`,`FacultyID`),

CONSTRAINT `FK\_COURSE\_DNumber`

FOREIGN KEY (`DNumber`) REFERENCES `DEPARTMENT` (`DNumber`),

CONSTRAINT `FK\_COURSE\_Faculty` FOREIGN KEY (`FacultyID`) REFERENCES `FACULTY` (`FacultyID`)

) ;



--Inserting data for table `COURSE`--

INSERT INTO `COURSE` (`CourseNo`, `CourseName`, `NumbrOfCredits`, `RegistrationDate`, `FacultyID`, `DNumber`, `Semester`) VALUES

('5CCI01', 'DATABASE MANAGEMENT SYSTEM', 4, '2018-08-03', 'CS04', 'CS', 5),

('5CCI02', 'DATA COMMUNICATIONS', 4, '2018-08-03', 'CS01', 'CS', 5),

('5CCI04', 'UNIX AND SHELL PROGRAMMING', 4, '2018-08-03', 'CS05', 'CS', 5),

('5ME01', 'DESIGN OF MACHINE ELEMENTS-1', 4.5, '2018-08-03', 'MC01', 'MC', 5),

('5ME02', 'TURBO MACHINERY', 3.5, '2018-08-03', 'MC02', 'MC', 5),

('5ME04', 'FINITE ELEMENT METHODS', 3.5, '2018-08-03', 'MC04', 'MC', 5),

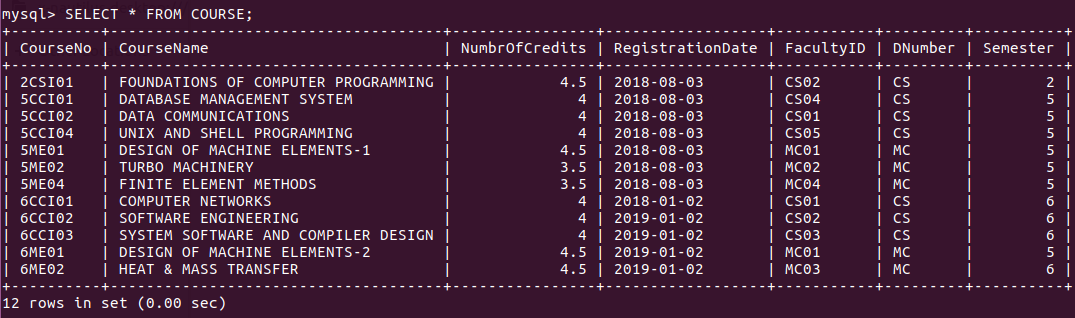
('6CCI01', 'COMPUTER NETWORKS', 4, '2018-01-02', 'CS01', 'CS', 6),

('6CCI02', 'SOFTWARE ENGINEERING', 4, '2019-01-02', 'CS02', 'CS', 6),

('6CCI03', 'SYSTEM SOFTWARE AND COMPILER DESIGN', 4, '2019-01-02', 'CS03', 'CS', 6),

('6ME01', 'DESIGN OF MACHINE ELEMENTS-2', 4.5, '2019-01-02', 'MC01', 'MC', 5),

('6ME02', 'HEAT & MASS TRANSFER', 4.5, '2019-01-02', 'MC03', 'MC', 6);



**Creating Table REGISTERED**

CREATE TABLE `REGISTERED` (

`StudentID` varchar(10) NOT NULL,

`CourseNo` varchar(20) NOT NULL,

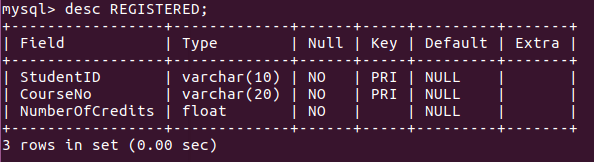
`NumberOfCredits` FLOAT,

PRIMARY KEY(StudentID,CourseNo),

FOREIGN KEY (`StudentID`)REFERENCES `STUDENT`(`StudentID`),

FOREIGN KEY(`FacultyID`)REFERENCES `FACULTY`(`FacultyID`)

)



mysql> show tables;

+---------------------------------------+

| Tables\_in\_subject\_management |

+---------------------------------------+

| COURSE |

| DEPARTMENT |

| FACULTY |

| REGISTERED |

| STUDENT |

+---------------------------------------+

5 rows in set (0.00 sec)

**QUERIES**

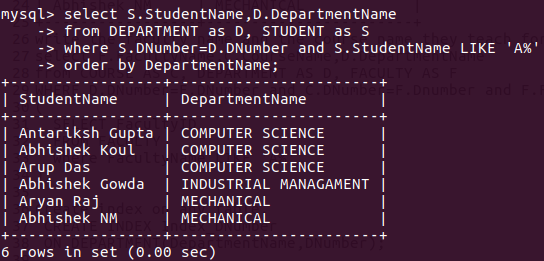
1. List the student names and the departments they belong to for student whose names start with letter ‘A’.

SELECT S.STUDENTNAME,D.DEPARTMENTNAME

FROM DEPARTMENT AS D, STUDENT AS S

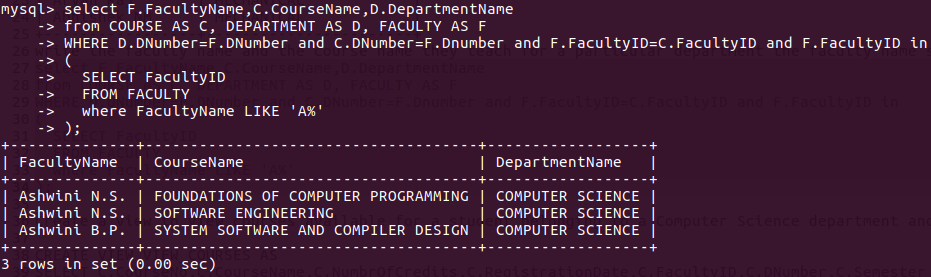
WHERE S.DNUMBER=D.DNUMBER AND S.STUDENTNAME LIKE 'A%'

ORDER BY DEPARTMENTNAME;



1. Write the faculty name and the course name they teach for a particular department the faculty name should start with an

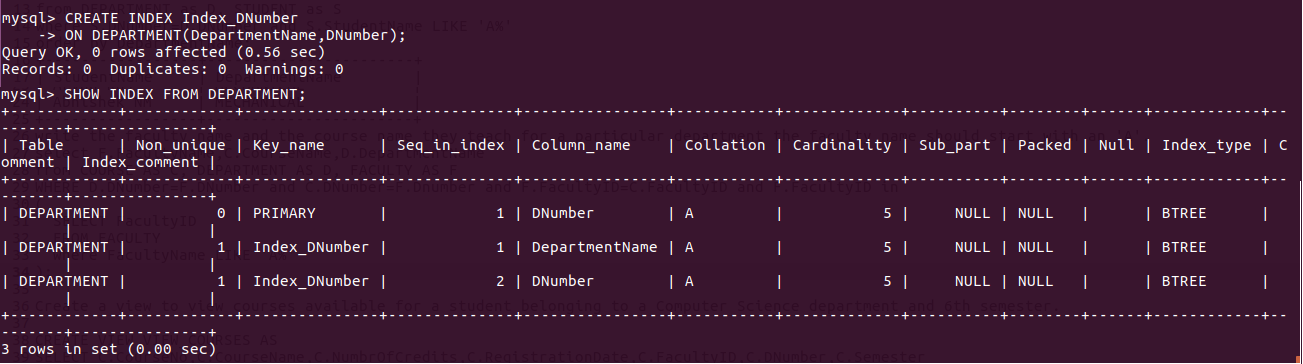
'A'.



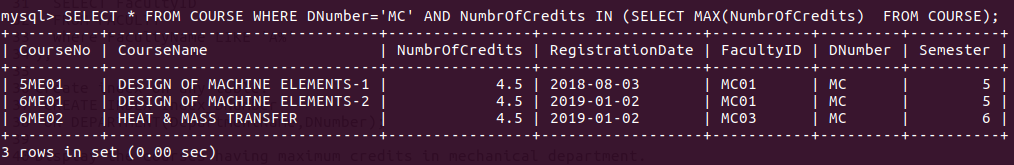
1. Create index on any table.

CREATE INDEX Index\_DNumber

ON DEPARTMENT(DepartmentName,DNumber);



4.) Display the courses having maximum credits in mechanical department.



5.) Create view to view courses available for a student belonging to a Computer Science department and 6th semester.

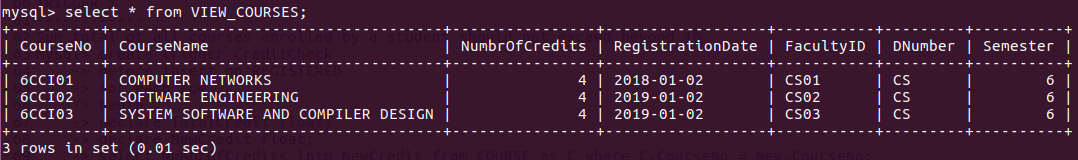
CREATE VIEW VIEW\_COURSES AS

SELECT C.CourseNo,C.CourseName,C.NumbrOfCredits,C.RegistrationDate,C.FacultyID,C.DNumber,C.Semester

FROM COURSE AS C

WHERE C.DNumber='CS' AND C.Semester=6

ORDER BY C.CourseName;



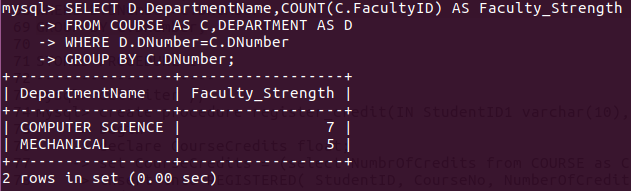
6.) Display every department and their respective faculty strength.

SELECT D.DepartmentName,COUNT(C.FacultyID) AS Faculty\_Strength

FROM COURSE AS C,DEPARTMENT AS D

WHERE D.DNumber=C.DNumber

GROUP BY C.DNumber;



**STORED PROCEDURE 1**

**-------------------------------------**

Write a stored procedure `register\_credit` which takes StudentID and CourseNo from the user and inserts the StudentID, CourseNo and NumberOfCredits allotted for the particular Course which may be obtained by querying the `COURSE` table.

delimiter //

create procedure register\_credit(IN StudentID1 varchar(10), IN CourseNo1 varchar(20))

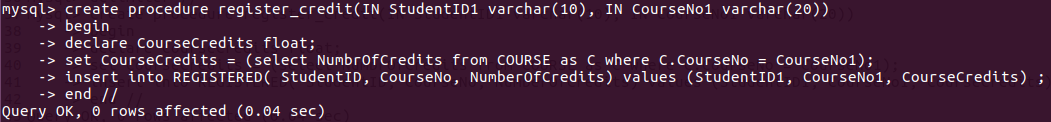
begin

declare CourseCredits float;

set CourseCredits = (select NumbrOfCredits from COURSE as C where C.CourseNo = CourseNo1);

insert into REGISTERED( StudentID, CourseNo, NumberOfCredits) values (StudentID1, CourseNo1, CourseCredits) ;

end //



**STORED PROCEDURE 2**

**------------------------------------**

Write a stored procedure to insert into FACULTY table only if it is a valid designation.

DELIMITER //

CREATE PROCEDURE VerifyFaculty(IN FacultyName1 VARCHAR(30),IN FacultyID1 VARCHAR(20),Designation1 VARCHAR(20),Salary1 float,DNumber1 VARCHAR(20))

BEGIN

IF (Designation1='Associate Professor' OR Designation1='Assistant Professor' OR Designation1='Professor' OR Designation1='HOD')

THEN

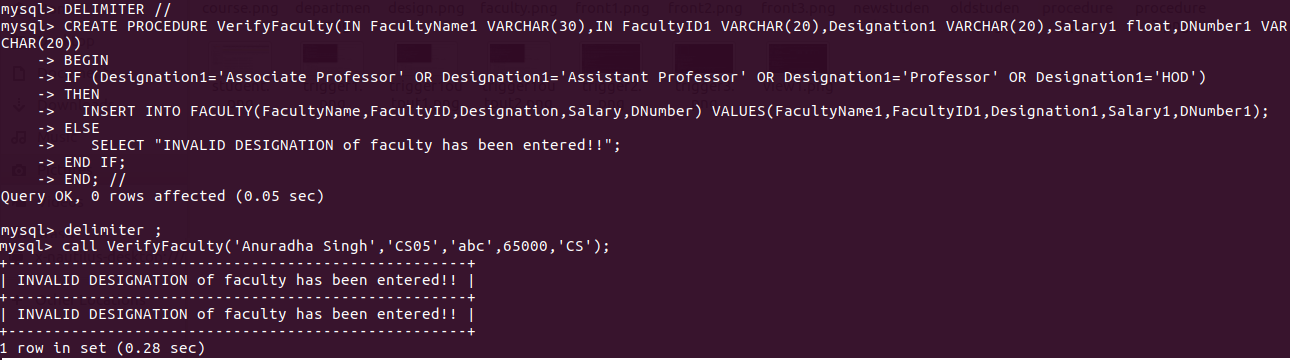
INSERT INTO FACULTY(FacultyName,FacultyID,Designation,Salary,DNumber) VALUES(FacultyName1,FacultyID1,Designation1,Salary1,DNumber1);

ELSE

SELECT "INVALID DESIGNATION of faculty has been entered!!";

END IF;

END; //



**TRIGGER 1**

**-------------------**

Sum total of all courses enrolled by a student should not exceed beyond 10.

DELIMITER //

create trigger CreditCheck

before insert on REGISTERED

for each row

begin

declare TotalCredit float;

declare NewCredit float;

select NumbrOfCredits into NewCredit from COURSE as C where C.CourseNo = New.CourseNo;

Select SUM(R.NumberOfCredits) into TotalCredit from REGISTERED as R where R.StudentID = NEW.StudentID;

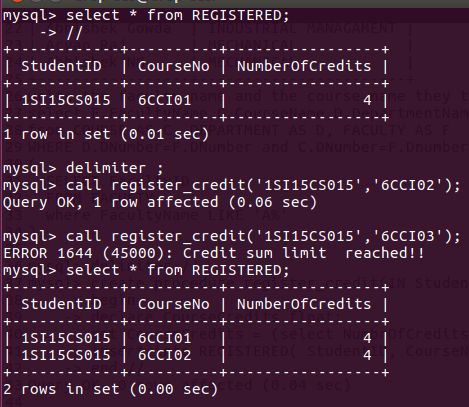
IF TotalCredit + NewCredit > 10

then

signal sqlstate '45000' set message\_text='Credit sum limit reached!! ';

end IF;

END; //



**TRIGGER 2**

**-------------------**

Number of credits for a given subject should not be less than 1

DELIMITER //

CREATE TRIGGER Course\_Credit\_Limit

BEFORE INSERT ON COURSE

FOR EACH ROW

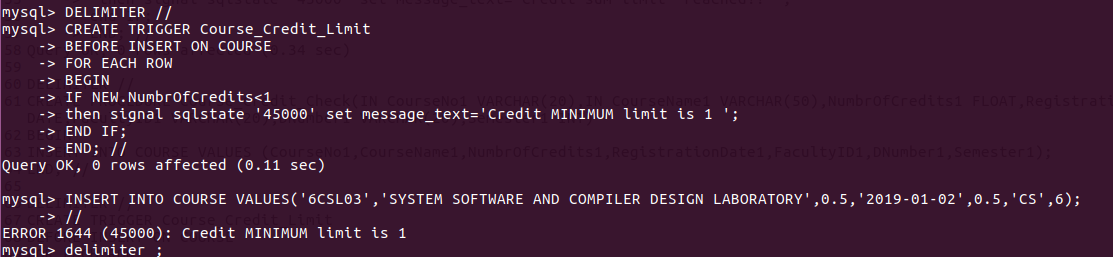
BEGIN

IF NEW.NumbrOfCredits<1

then signal sqlstate '45000' set message\_text='Credit MINIMUM limit is 1 ';

END IF;

END; //



**TRIGGER 3**

**---------------------**

A Faculty cannot teach more than 2 courses.

DELIMITER //

CREATE TRIGGER Faculty\_Course\_Limit

BEFORE INSERT ON COURSE

FOR EACH ROW

BEGIN

declare FacultyCount int;

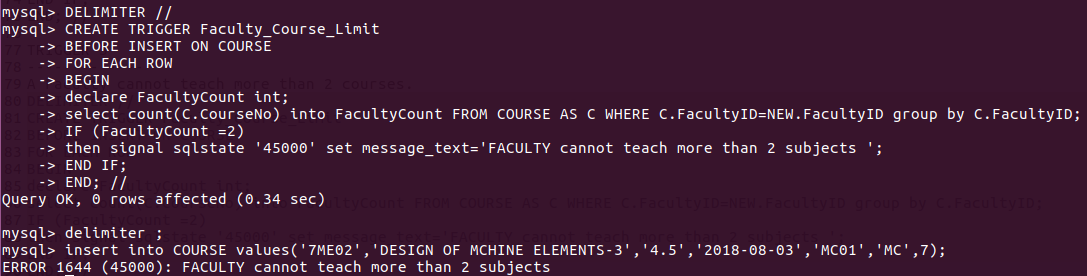
select count(C.CourseNo) into FacultyCount FROM COURSE AS C WHERE C.FacultyID=NEW.FacultyID group by C.FacultyID;

IF (FacultyCount =2)

then signal sqlstate '45000' set message\_text='FACULTY cannot teach more than 2 subjects ';

END IF;

END; //



**TRIGGER 4**

**--------------------**

Write a trigger to Validate a StudentID.

DELIMITER //

CREATE TRIGGER Verify\_StudentID

BEFORE INSERT ON STUDENT

FOR EACH ROW

BEGIN

IF (NEW.StudentID NOT LIKE '1SI[0-9][0-9][A-Z][A-Z][0-9][0-9][0-9]')

then signal sqlstate '45000' set message\_text='INVALID StudentID ';

END IF;

END; //

