

|  |  |  |  |
| --- | --- | --- | --- |
| **Academic Year** | **Module** | **Assessment Number** | **Assessment Type** |
| S20 | DipIT07 - Introduction To Database System | A1 | Individual Report |

School Management System

Student Id : NP03A190103

Student Name : Prashant Phuyal

Section : DC06

Module Leader : Deepson Shrestha

Lecturer : Deepson Shrestha

Submitted on : 29/05/2020

Acknowledgement

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

The success and final outcome of this project required a lot of guidance and assistance and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank Mr. Deepson Shrestha, our module Leader for providing me an opportunity to do the report work and giving me all support and guidance. Also, I want to thank Mr. Sachit Tandukar and Mr. Subiran Shrestha Sir who helped me complete the project. I am extremely thankful to Herald College and teachers for providing such a great opportunity, support, guidance and providing all the necessary information of Course Work.

Table of Contents

[I. System Description 1](#_Toc41501348)

[A. MySQL 1](#_Toc41501349)

[B. Database Description 1](#_Toc41501350)

[II. Data Dictionaries 2](#_Toc41501351)

[III. ER Diagram 10](#_Toc41501352)

[IV. Create Databases and Tables / Insert sufficient Data 11](#_Toc41501353)

[V. Select Statements using Different Functions 28](#_Toc41501354)

[VI. Select Statements using Sub Query 38](#_Toc41501355)

[VII. Select Statements using Count and Group Functions 40](#_Toc41501356)

[VIII. Select Statements using Different Joins 41](#_Toc41501357)

[IX. Insert Statement 45](#_Toc41501358)

[X. Update Statement 48](#_Toc41501359)

[XI. Delete Statement 50](#_Toc41501360)

[XII. Normalization 53](#_Toc41501361)

[A. First Normal Form 53](#_Toc41501362)

[B. Second Normal Form 54](#_Toc41501363)

[C. Third Normal Form 56](#_Toc41501364)

[Conclusion 57](#_Toc41501365)

[References 58](#_Toc41501366)

# System Description

## MySQL

MySQL is an open source relational database management system (RDBMS) with a client- server model. RDBMS is a software or service used to create and manage databases based on a relational model. A database is an organized collection of various forms of data. It is also known as a structured set of data that is accessible in many ways through computer.

The aim of this case study is to design and develop a database for the school to manage the record of students, classes, courses, subjects, teachers and books in the school. It also maintain the records of students assigned in class, staff departments, members of library, subject in courses, examination given by students, subject marks etc. in MySQL database.

## Database Description

School management system is an information system to manage school related data and day to day activities. In school, there are students, teachers, Courses, Staff, library, Departments, classes and subjects etc. There are courses where students enrolls, records of parent identified by students, classes for courses, attendance of students of each classes, subjects inside each courses which is taught by specific teachers, teachers categorized to specific department as per their subjects, records of staffs in school who works in different fields, teaching hours of each subjects per day, Examination which is to be held per year and marks scored by students in each subjects. Also, there is a Library in school where there are many books which can be issued by the members of library, members can either issue zero or more than one books as per their choice.

All the records are store in their own specific tables, for example there is a table for Members and books where record of members are kept in members table and record of books are kept in Books table respectively and to relate them together there is a table named, issued books through which library manager will able to know which books are issued by which members easily.

In this way, data can be secured, organized, managed and accessed whenever it is required which in turn improve the reliability of schools information.

# Data Dictionaries

Following are the tables along with constraints used in School Management System database.

A. School Table

This table consists of information of school name, its address and CST No. of school.

*Constraints*: Data can’t be empty in this table, and there is no foreign key and primary key. BIGINT is used so that large number can be added in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| school\_name | VARCHAR | 60 | NOT NULL | Name of school |
| school\_address | VARCHAR | 60 | NOT NULL | School address |
| CST No. | BIGINT | - | NOT NULL | School SCT No. |

B. Student Table

This table consists of all the necessary information of the students like student name, date of birth, Student roll number, student id and gender.

*Constraints*: Student id is used as primary key and Class id is used as Foreign key which so that class table references student table for classes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| student\_id | VARCHAR | 10 | PK | Student id |
| student\_name | VARCHAR | 30 | NOT NULL | Name of Student |
| Class\_id | VARCHAR | 10 | FK | ID of class where student study |
| roll\_no | INT | 11 | NOT NULL | Student Roll No. |
| DOB | DATE | - | NOT NULL | Date of Birth |
| gender | VARCHAR | 20 | NOT NULL | Student Gender |

C. Parent Table

This table stores the detail of parents which consists attribute like parent name, phone number and parent address.

*Constraints*: Student ID is used as a foreign key to reference the student table and BIGINT is used to store large Integer character.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| parent\_Name | VARCHAR | 30 | NOT NULL | Parent’s Name |
| student\_id | VARCHAR | 10 | FK | Student Id |
| phone\_number | BIGINT | - | NOT NULL | Parent’s Phone No. |
| parent\_address | VARCHAR | 40 | NOT NULL | Parents Address |

D. Course Table

This table stores the courses of education school provides and student can choose the courses as per their need.

*Constraints*: Course ID is primary key here and corresponding course id should be in subject table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| course\_id | VARCHAR | 10 | PK | Course ID |
| course\_name | VARCHAR | 20 | NOT NULL | Name of Course |

E. Attendance Table

This table stores the data about the total numbers of study days student were present

*Constraints*: This table consists of attribute as total\_class and total\_present and student \_d is a foreign key to reference the table, student.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| student\_id | VARCHAR | 10 | FK | Student ID |
| total\_class | INT | 11 | NOT NULL | Total study class |
| total\_present | INT | 11 | NOT NULL | Total present days |

F. Class Table

This table stores the total number of classes in the school and student table reference this table to keep the record of the student’s class

*Constraints*: This table consists of attribute as class id as a primary key and class \_name student are associated with.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| class\_id | VARCHAR | 10 | PK | Class ID |
| class\_name | VARCHAR | 20 | NOT NULL | Name of Class |

G. Subject Table

Subject table holds the data of the subject as per the courses and classes.

*Constraints*: This table consists of attribute as subject id as a primary key and subject name as name of the subject. Class id and course id references class table and course table and relation between them.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| subject\_id | VARCHAR | 10 | PK | Subject ID |
| subject\_name | VARCHAR | 20 | NOT NULL | Subject Name |
| class\_id | VARCHAR | 10 | FK | Class ID |
| course\_id | VARCHAR | 10 | FK | Course ID |

H. Period Hour Table

This table consists of information related the time period of the each subject.

*Constraints*: This table has attribute as subject id as a foreign key which references subject table in order to know each subsect marks. FLOAT is used to store decimal value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| subject\_id | VARCHAR | 10 | FK | Subject ID |
| time\_period | FLOAT | - | NOT NULL | Time in hour |

1. Teacher Table

This table consists of all the necessary information of the teachers like teacher name, teacher address.

*Constraints*: Teacher id is used as primary key to uniquely identity the teacher and Department id is used as foreign key to find which department teachers is associated with.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| teacher\_id | INT | 11 | PK | Teacher ID |
| teacher\_name | VARCHAR | 30 | NOT NULL | Teacher Name |
| teacher\_address | VARCHAR | 40 | NOT NULL | Teacher Address |
| department\_id | VARCHAR | 10 | FK | Teacher belonging Department |

J. Department Table

This table holds the data for department name and department id.

*Constraints*: Here, department id is primary key which is to be referenced on Teacher and Staff table respectively.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| department\_id | VARCHAR | 10 | PK | Department ID |
| department\_name | VARCHAR | 50 | NOT NULL | Department Name |

K. Marks Table

In this table marks of each subject and their respective subject is recorded.

*Constraints*: This table have two foreign key i.e. student id and subject id referenced by student table and subject table respectively and attribute sub marks holds the marks of subject.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| student\_id | VARCHAR | 10 | FK | Student ID |
| subject\_id | VARCHAR | 10 | FK | Subject ID |
| sub\_marks | INT | 11 | NOT NULL | Marks of each subject of students |

L. Exam Table

This table consists of data regarding the exam and the date in which subject exam has to be executed.

*Constraints*: This table has primary key exam id which identifies this table and which is to be referenced in exam\_subject table for relation between subject table and exam table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| exam\_id | VARCHAR | 10 | PK | Exam ID |
| exam\_date | DATE | - | NOT NULL | Date of exam for each subjects |

M. Staff Table

This table consists of all the necessary information of the Staff like staff name, work role staff has to do, Staff phone number and department id where staff is associated.

Constraints: Here staff id is primary key which uniquely identifies records and department id is foreign key which references department table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| staff\_id | VARCHAR | 10 | PK | Exam ID |
| staff\_name | VARCHAR | 30 | NOT NULL | Staff Name |
| work\_role | VARCHAR | 40 | NOT NULL | Task staff need to do |
| staff\_phone | BIGINT | - | NOT NULL | Staff Phone number |
| department\_id | VARCHAR | 10 | FK | Department ID |

N. Library Table

This table stores information of library, its name and id.

*Constraints*: This table has library id as primary key, it’s not referenced in any table but added incase student as more than one library.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| library\_id | INT | 11 | PK | Library ID |
| library\_name | VARCHAR | 40 | NOT NULL | Library Name |

O. Books Table

The books present in the library is added or inserted into this table, which stores the name of book and writer’s name.

*Constraints*: There is auto incremented book id which starts from 1000 and uniquely identifies record of books, book id is used as foreign key in table (issued books) as a relation between member table and books table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| book\_id | INT | 11 | PK | Book ID, Auto Increment |
| book\_name | VARCHAR | 40 | NOT NULL | Book Name |
| writer | VARCHAR | 30 | NOT NULL | Book Writer Name |

P. Members Table

This table consists of member id and student id

*Constraints*: student id as a foreign key works to provide details of the members referenced by student table. Primary key member id is used as foreign key to reference members table in issued books table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| member\_id | VARCHAR | 10 | PK | Member ID |
| student\_id | VARCHAR | 10 | FK | Student ID |

Q. issued\_books table

This table issued books is a relation between two table members and book.

*Constraints*: Two foreign key of member id and book id references member and book table respectively, null value is not accepted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| member\_id | VARCHAR | 10 | FK | Member ID |
| book\_id | INT | 11 | FK | Book ID |

R. student\_course Table

This table student\_course is a relation between two table student and course.

*Constraints*: Two foreign key of course id and student id references course and student table respectively, null value is not accepted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| course\_id | VARCHAR | 10 | FK | Course ID |
| student\_id | VARCHAR | 10 | FK | Student ID |

S. exam\_subject Table

This table exam\_subject is a relation between two table exam and table.

*Constraints*: Two foreign key of exam id and subject id references exam and subject table respectively, null value is not accepted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| exam\_id | VARCHAR | 10 | FK | Exam ID |
| subject\_id | VARCHAR | 10 | FK | Subject ID |

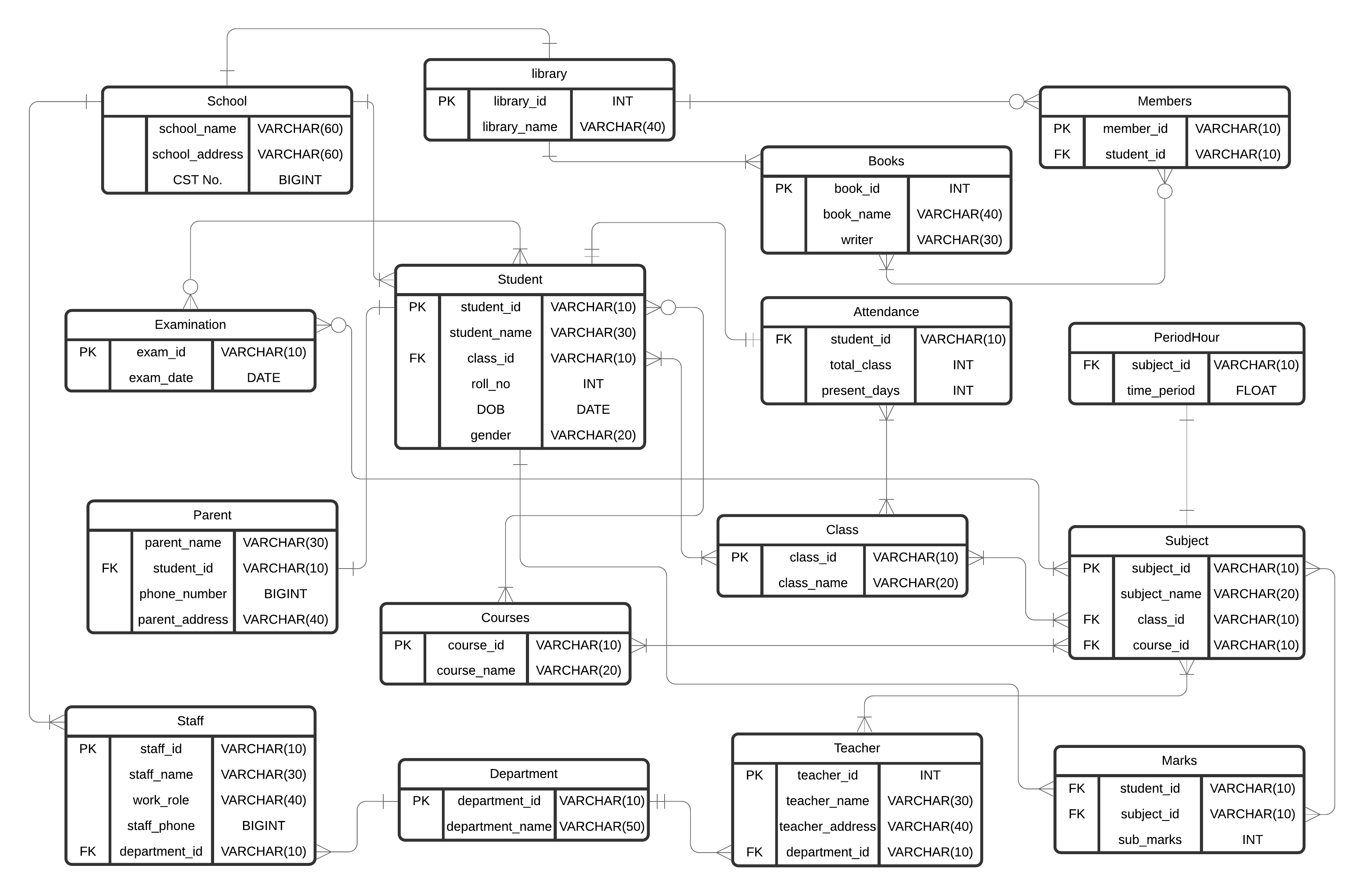
T. teacher\_teaches Table

This table teacher\_teaches is a relation between two table teacher and subject.

*Constraints*: Two foreign key of teacher id and subject id references teacher and subject table respectively, null value is not accepted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| teacher\_id | INT | 11 | FK | Teacher ID |
| subject\_id | VARCHAR | 10 | FK | Subject ID |

# ER Diagram



# Create Databases and Tables / Insert sufficient Data

A. Student Table

CREATE TABLE `Student` (

`student\_id` VARCHAR(10),

`student\_name` VARCHAR(30) NOT NULL,

`class\_id` VARCHAR(10) NOT NULL,

`roll\_no` INT NOT NULL,

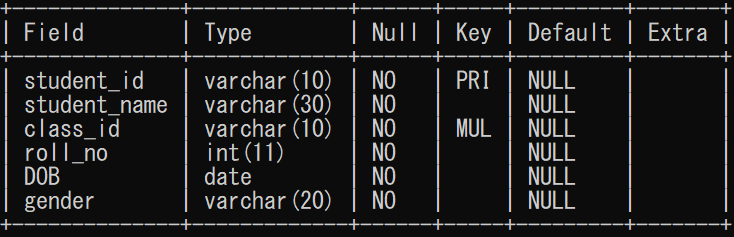
`DOB` DATE NOT NULL,

`gender` VARCHAR(20) NOT NULL,

PRIMARY KEY (`student\_id`)

FOREIGN KEY (`class\_id`) REFERENCES Class(`class\_id`);

);



INSERT INTO `Student` VALUES

('NP01', 'Prashant Phuyal', 'LT01', '10', '2001-01-16', 'MALE'),

('NP02', 'Razz BC', 'LT01', '11', '2001-02-11', 'MALE'),

('NP03', 'Aayush Shrestha', 'LT01', '12', '2001-02-19', 'MALE'),

('NP04', 'Rachana Subedi', 'LT02', '09', '2001-05-16', 'FEMALE'),

('NP05', 'Aavinab Shah', 'LT02', '14', '2001-07-11', 'MALE'),

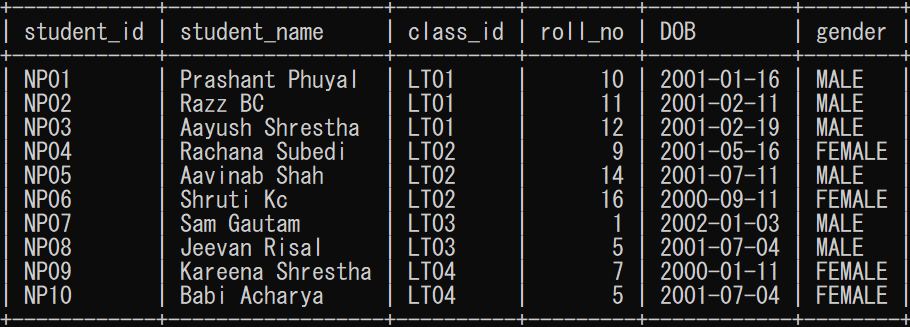
('NP06', 'Shruti Kc', 'LT02', '16', '2000-09-11', 'FEMALE'),

('NP07', 'Sam Gautam', 'LT03', '01', '2002-01-03', 'MALE'),

('NP08', 'Jeevan Risal', 'LT03', '05', '2001-07-04', 'MALE'),

('NP09', 'Kareena Shrestha', 'LT04', '07', '2000-01-11', 'FEMALE'),

('NP10', 'Babi Acharya', 'LT04', '05', '2001-07-04', 'FEMALE');



B. Parent Table

CREATE TABLE `Parent` (

`parent\_name` VARCHAR(30) NOT NULL,

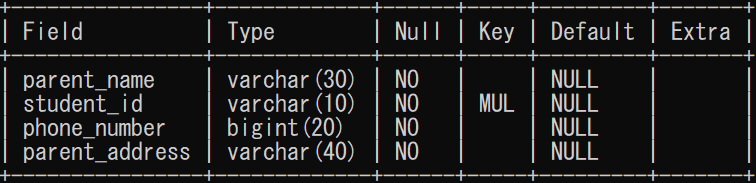
`student\_id` VARCHAR(10) NOT NULL,

`phone\_number` BIGINT NOT NULL,

`parent\_address` VARCHAR(40) NOT NULL,

FOREIGN KEY (`student\_id`) REFERENCES Student(`student\_id`)

);



INSERT INTO `Parent` VALUES

('Chandra', 'NP01', '9814365499', 'Itahari-9, Sunsari'),

('Calvin', 'NP02', '9814262492', 'Dharan-17, Sunsari'),

('Jonathan', 'NP03', '9826245349', 'Biratnagar-1, Morang'),

('Tristan', 'NP04', '9812412124', 'Biratnagar-2, Morang'),

('Bran', 'NP05', '9868758782', 'Itahari-1, Sunsari'),

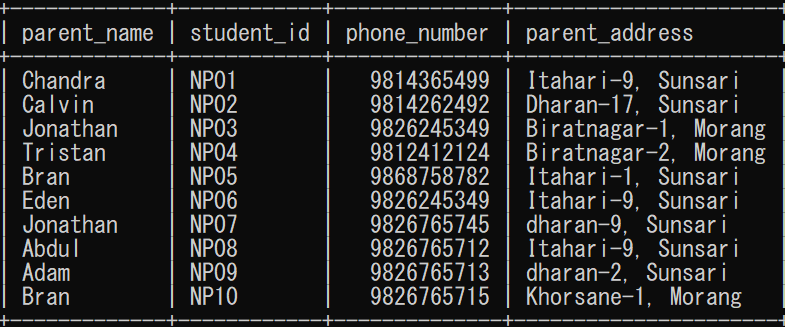
('Eden', 'NP06', '9826245349', 'Itahari-9, Sunsari'),

('Jonathan', 'NP07', '9826765745', 'dharan-9, Sunsari'),

('Abdul', 'NP08', '9826765712', 'Itahari-9, Sunsari'),

('Adam', 'NP09', '9826765713', 'dharan-2, Sunsari'),

('Bran', 'NP10', '9826765715', 'Khorsane-1, Morang');



C. Attandance Table

CREATE TABLE `Attendance` (

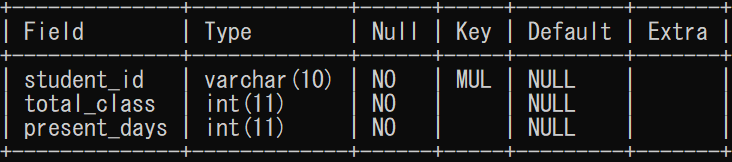
`student\_id` VARCHAR(10) NOT NULL,

`total\_class` INT NOT NULL,

`present\_days` INT NOT NULL,

FOREIGN KEY (`student\_id`) REFERENCES Student(`student\_id`)

);



INSERT INTO `Attendance` VALUES

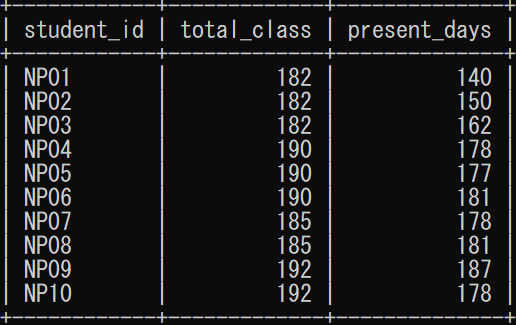
('NP01', '182', '140'), ('NP02', '182','150'),

('NP03', '182','162'), ('NP04', '190','178'),

('NP05', '190','177'), ('NP06', '190','181'),

('NP07', '185','178'), ('NP08', '185','181'),

('NP09', '192','187'), ('NP10', '192','178');



D. Courses Table

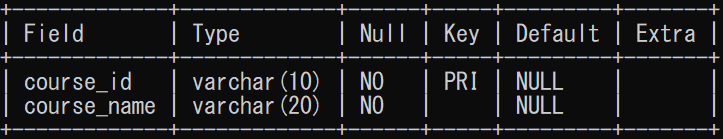
CREATE TABLE `Courses` (

`course\_id` VARCHAR(10),

`course\_name` VARCHAR(20) NOT NULL,

PRIMARY KEY (`course\_id`)

);



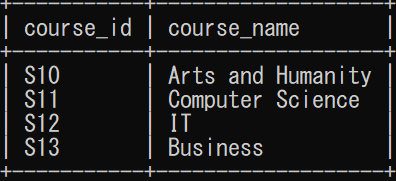
INSERT INTO `Courses` VALUES

('S10', 'Arts and Humanity'),

('S11', 'Computer Science'),

('S12', 'IT'),

('S13', 'Business');



E. Class Table

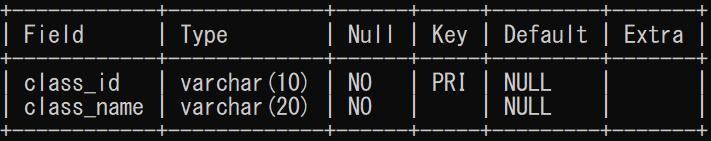
CREATE TABLE `Class` (

`class\_id` VARCHAR(10),

`class\_name` VARCHAR(20) NOT NULL,

PRIMARY KEY (`class\_id`)

);



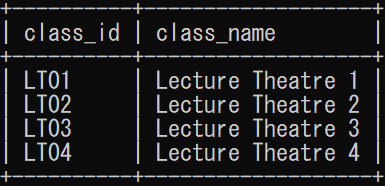
INSERT INTO `class` VALUES

('LT01','Lecture Theatre 1'),

('LT02','Lecture Theatre 2'),

('LT03','Lecture Theatre 3'),

('LT04','Lecture Theatre 4');



F. Subject Table

CREATE TABLE `Subject` (

`subject\_id` VARCHAR(10),

`subject\_name` VARCHAR(20) NOT NULL,

`class\_id` VARCHAR(10) NOT NULL,

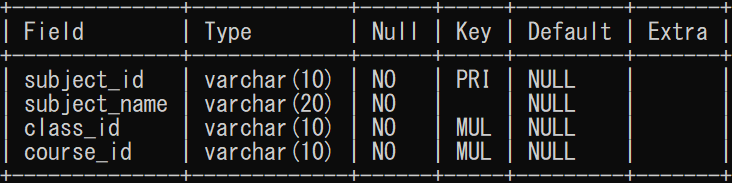
`course\_id` VARCHAR(10) NOT NULL,

PRIMARY KEY (`subject\_id`),

FOREIGN KEY (`class\_id`) REFERENCES Class(`class\_id`),

FOREIGN KEY (`course\_id`) REFERENCES Courses(`course\_id`)

);



INSERT INTO `Subject` VALUES

('E01', 'History', 'LT01','S10'),

('E02', 'Music and Art','LT01','S10'),

('E03', 'Philosophy', 'LT01','S10'),

('F01', 'Algorithms', 'LT02','S11'),

('F02', 'Software Development', 'LT02','S11'),

('F03', 'Computer Security', 'LT02','S11'),

('F04', 'Design and Product', 'LT02','S11'),

('H10', 'Networking', 'LT03','S12'),

('H11', 'Cloud Computing', 'LT03','S12'),

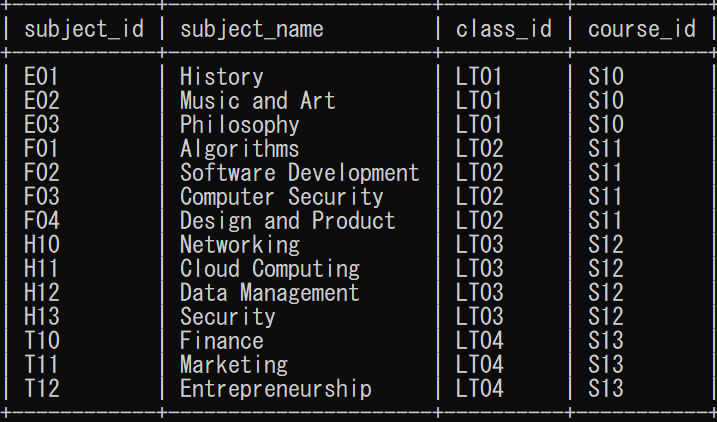
('H12', 'Data Management', 'LT03','S12'),

('H13', 'Security', 'LT03','S12'),

('T10', 'Finance', 'LT04','S13'),

('T11', 'Marketing', 'LT04','S13'),

('T12', 'Entrepreneurship', 'LT04','S13');



G. PeriodHour Table

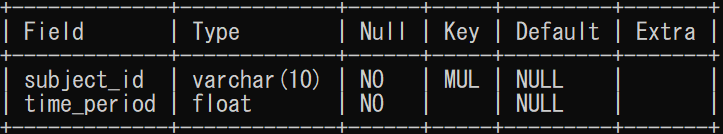
CREATE TABLE `PeriodHour` (

`subject\_id` VARCHAR(10) NOT NULL,

`time\_period` FLOAT NOT NULL,

FOREIGN KEY (`subject\_id`) REFERENCES Subject(`subject\_id`)

);

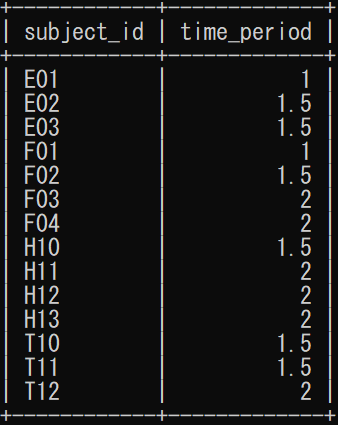


INSERT INTO `PeriodHour` VALUES

('E01', '1'), ('E02', '1.5'), ('E03','1.5'), ('F01', '1'), ('F02', '1.5'),

('F03', '2'), ('F04', '2'), ('H10', '1.5'), ('H11', '2'), ('H12', '2'),

('H13', '2'), ('T10', '1.5'), ('T11','1.5'), ('T12', '2');



H. Marks Table

CREATE TABLE `Marks` (

`student\_id` VARCHAR(10) NOT NULL,

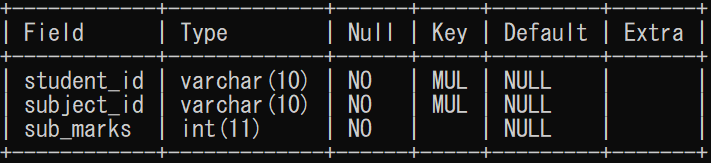
`subject\_id` VARCHAR(10) NOT NULL,

`sub\_marks` INT NOT NULL,

FOREIGN KEY (`student\_id`) REFERENCES Student(`student\_id`),

FOREIGN KEY (`subject\_id`) REFERENCES Subject(`subject\_id`)

);



INSERT INTO `Marks` VALUES

('NP01','E01','60'), ('NP01','E02','65'), ('NP01','E03','75'),

('NP02','E01','63'), ('NP02','E02','55'), ('NP02','E03','61'),

('NP03','E01','70'), ('NP03','E02','61'), ('NP03','E03','65'),

('NP04','F01','60'), ('NP04','F02','65'), ('NP04','F03','57'), ('NP04','F04','70'),

('NP05','F01','61'), ('NP05','F02','67'), ('NP05','F03','75'), ('NP05','F04','75'),

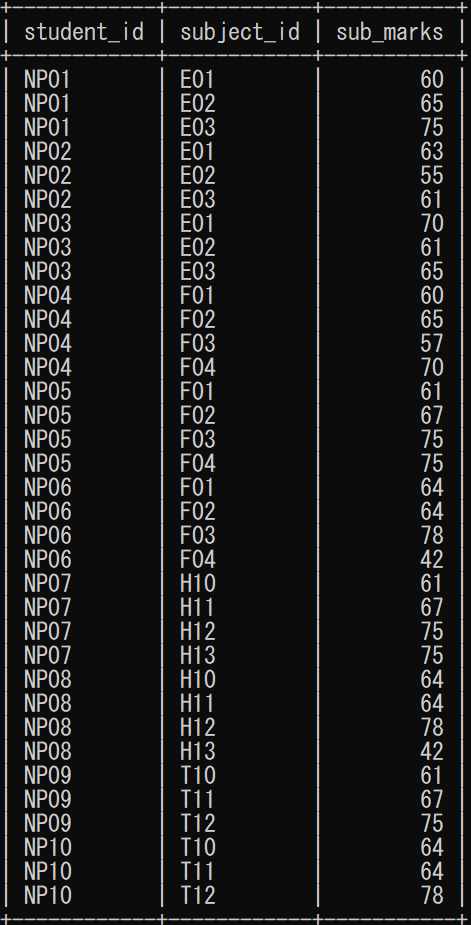
('NP06','F01','64'), ('NP06','F02','64'), ('NP06','F03','78'), ('NP06','F04','42'),

('NP07','H10','61'), ('NP07','H11','67'), ('NP07','H12','75'), ('NP07','H13','75'),

('NP08','H10','64'), ('NP08','H11','64'), ('NP08','H12','78'), ('NP08','H13','42'),

('NP09','T10','61'), ('NP09','T11','67'), ('NP09','T12','75'),

('NP10','T10','64'), ('NP10','T11','64'), ('NP10','T12','78');



I. Department Table

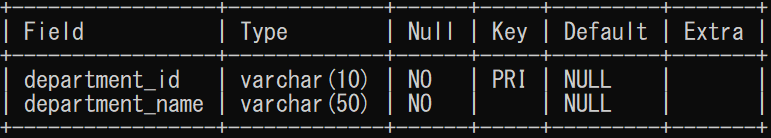
CREATE TABLE `Department` (

`department\_id` VARCHAR(10),

`department\_name` VARCHAR(50) NOT NULL,

PRIMARY KEY (`department\_id`)

);



INSERT INTO `Department` VALUES

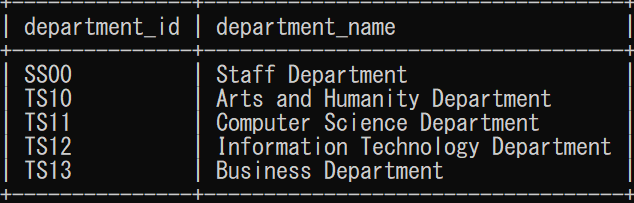
('TS10','Arts and Humanity Department'),

('TS11','Computer Science Department'),

('TS12','Information Technology Department'),

('TS13','Business Department'),

('SS00','Staff Department');



J. Teacher Table

CREATE TABLE `Teacher` (

`teacher\_id` INT,

`teacher\_name` VARCHAR(30) NOT NULL,

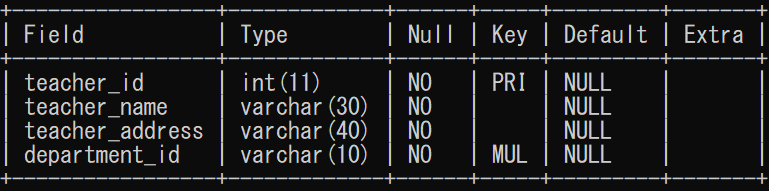
`teacher\_address` VARCHAR(40) NOT NULL,

`department\_id` VARCHAR(10) NOT NULL,

PRIMARY KEY (`teacher\_id`),

FOREIGN KEY (`department\_id`) REFERENCES Department(`department\_id`)

);



INSERT INTO `Teacher` VALUES

('100','Arun Knott','Pokhara','TS10'),

('102',' Sufyan Wilder', 'Hetauda', 'TS10'),

('104',' Marta Rangel','Biratnagar','TS11'),

('105',' Malcolm Lucas', 'Itahari', 'TS11'),

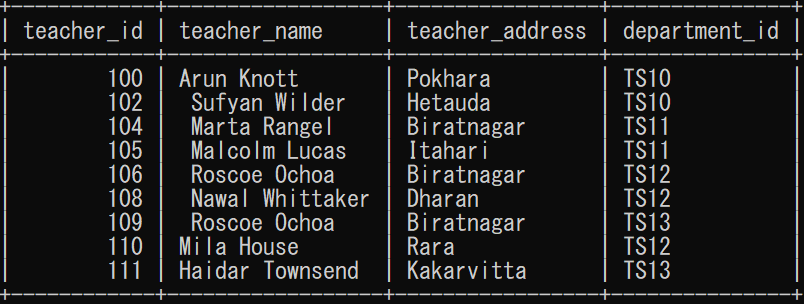
('106',' Roscoe Ochoa','Biratnagar','TS12'),

('108',' Nawal Whittaker', 'Dharan', 'TS12'),

('110', 'Mila House', 'Rara', 'TS12'),

('109',' Roscoe Ochoa','Biratnagar','TS13'),

('111','Haidar Townsend', 'Kakarvitta', 'TS13');



K. Staff Table

CREATE TABLE `Staff` (

`staff\_id` VARCHAR(10),

`staff\_name` VARCHAR(30) NOT NULL,

`work\_role` VARCHAR(40) NOT NULL,

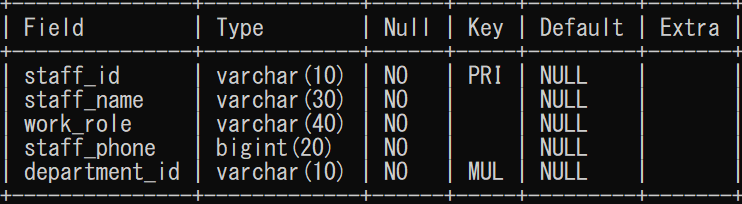
`staff\_phone` BIGINT NOT NULL,

`department\_id` VARCHAR(10) NOT NULL,

PRIMARY KEY (`staff\_id`),

FOREIGN KEY (`department\_id`) REFERENCES Department(`department\_id`)

);



INSERT INTO `Staff` VALUES

('SF01','Romany Firth','Accountant','9865432411','SS00'),

('SF02','Anna Lowe', 'Accountant Assistance','9862353212','SS00'),

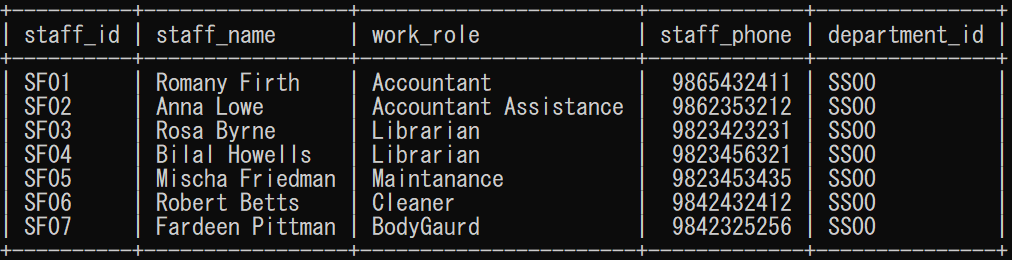
('SF03','Rosa Byrne','Librarian','9823423231','SS00'),

('SF04','Bilal Howells','Librarian','9823456321','SS00'),

('SF05','Mischa Friedman','Maintanance','9823453435','SS00'),

('SF06','Robert Betts','Cleaner','9842432412','SS00'),

('SF07','Fardeen Pittman','BodyGaurd','9842325256','SS00');



L. Eximination Table

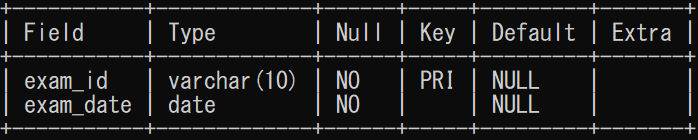
CREATE TABLE `Eximination` (

`exam\_id` VARCHAR(10),

`exam\_date` DATE NOT NULL,

PRIMARY KEY (`exam\_id`)

);



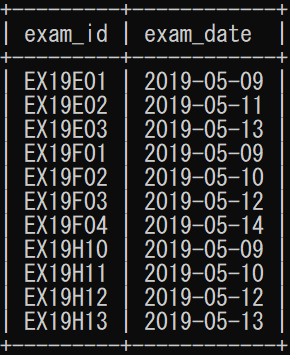
INSERT INTO `Eximination` VALUES

('EX19E01','2019-05-09'), ('EX19E02','2019-05-11'), ('EX19E03','2019-05-13'),

('EX19F01','2019-05-09'), ('EX19F02','2019-05-10'), ('EX19F03','2019-05-12'),

('EX19F04','2019-05-14'), ('EX19H10','2019-05-09'), ('EX19H11','2019-05-10'),

('EX19H12','2019-05-12'), ('EX19H13','2019-05-13');



M. Library Table

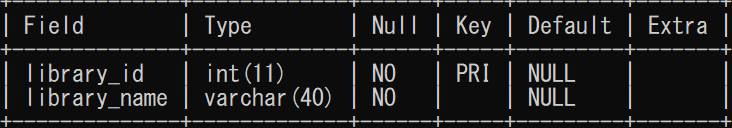
CREATE TABLE `Library` (

`library\_id` INT,

`library\_name` VARCHAR(40) NOT NULL,

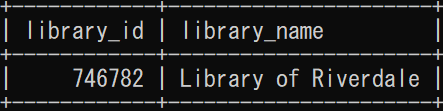
PRIMARY KEY (`library\_id`)

);



INSERT INTO `Library` VALUES

('746782','Library of Riverdale');



N. School Table

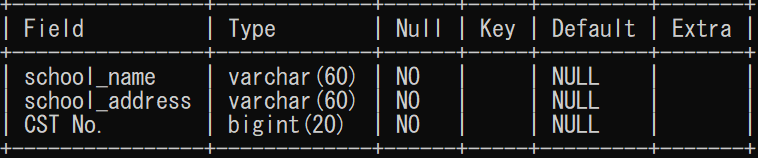
CREATE TABLE `School` (

`school\_name` VARCHAR(60) NOT NULL,

`school\_address` VARCHAR(60) NOT NULL,

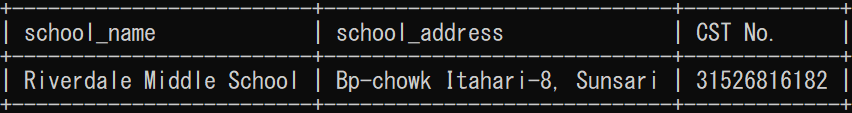
`CST No.` BIGINT NOT NULL

);



INSERT INTO `School` VALUES

('Riverdale Middle School', 'Bp-chowk Itahari-8, Sunsari', '31526816182');



O. Members Table

CREATE TABLE `Members` (

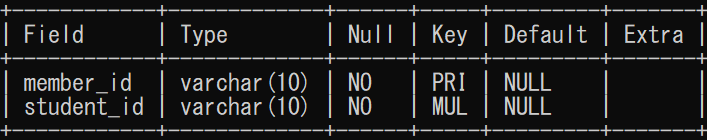
`member\_id` VARCHAR(10),

`student\_id` VARCHAR(10) NOT NULL,

PRIMARY KEY (`member\_id`),

FOREIGN KEY (`student\_id`) REFERENCES Student(`student\_id`)

);

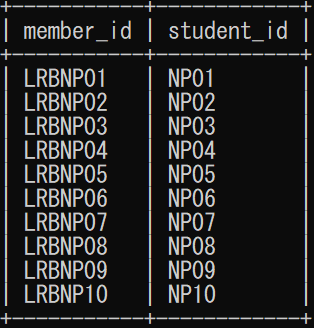


INSERT INTO `Members` VALUES

('LRBNP01','NP01'), ('LRBNP02','NP02'), ('LRBNP03','NP03'), ('LRBNP04','NP04'),

('LRBNP05','NP05'), ('LRBNP06','NP06'), ('LRBNP07','NP07'), ('LRBNP08','NP08'),

('LRBNP09','NP09'), ('LRBNP10','NP10');



P. Books Table

CREATE TABLE `Books` (

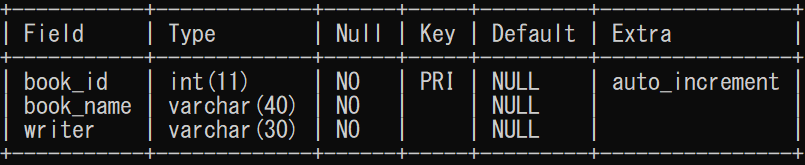
`book\_id` INT AUTO\_INCREMENT,

`book\_name` VARCHAR(40) NOT NULL,

`writer` VARCHAR(30) NOT NULL,

PRIMARY KEY (`book\_id`)

);



INSERT INTO `Books` VALUES

('1000', 'To Kill a Mockingbird', 'Harper Lee'),

('1001', '1984', 'George Orwell'),

('1002', 'Harry Potter and the Philosopher\'s Stone', 'J.K. Rowling'),

('1003', 'the Lord of the Rings', 'J.R.R'),

('1004', 'the Great Gatsby', 'F. Scott Fitzgerald'),

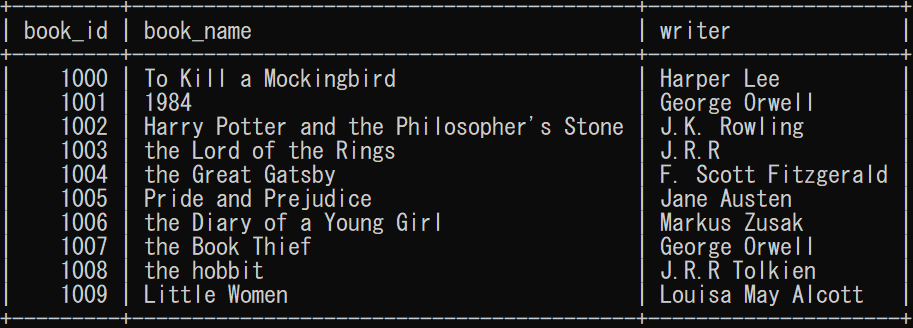
('1005', 'Pride and Prejudice', 'Jane Austen'),

('1006', 'the Diary of a Young Girl', 'Markus Zusak'),

('1007', 'the Book Thief', 'George Orwell'),

('1008', 'the hobbit', 'J.R.R Tolkien'),

('1009', 'Little Women', 'Louisa May Alcott');



Q. issued\_books Table

CREATE TABLE `issued\_books` (

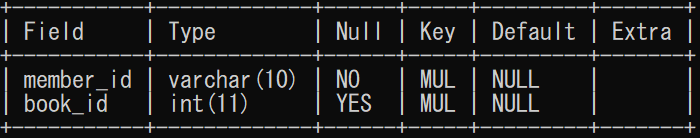
`member\_id` VARCHAR(10) NOT NULL,

`book\_id` INT ,

FOREIGN KEY (`member\_id`) REFERENCES Members(`member\_id`),

FOREIGN KEY (`book\_id`) REFERENCES Books(`book\_id`)

);

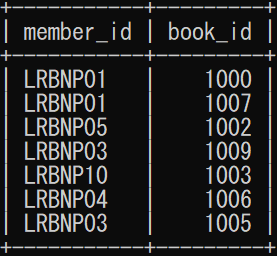


INSERT INTO `issued\_books` VALUES

('LRBNP01','1000'), ('LRBNP01','1007'), ('LRBNP05','1002'),

('LRBNP03','1009'), ('LRBNP10','1003'), ('LRBNP04','1006'),

('LRBNP03','1005');



R. teacher\_teaches Table

CREATE TABLE `teacher\_teaches` (

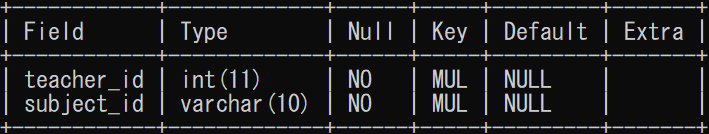
`teacher\_id` INT NOT NULL,

`subject\_id` VARCHAR(10) NOT NULL,

FOREIGN KEY (`teacher\_id`) REFERENCES Teacher(`teacher\_id`),

FOREIGN KEY (`subject\_id`) REFERENCES Subject(`subject\_id`)

);



INSERT INTO `teacher\_teaches` VALUES

('100','E01'), ('100','E03'),

('102','E02'), ('102','E01'),

('104','F01'), ('104','F04'),

('105','F02'), ('105','F03'),

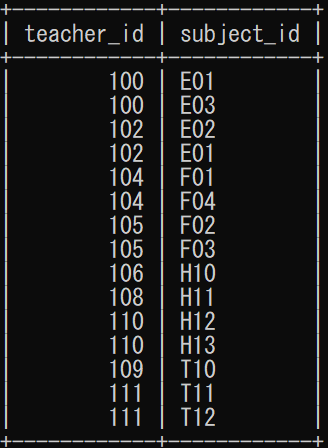
('106','H10'),

('108','H11'),

('110','H12'), ('110','H13'),

('109','T10'),

('111','T11'), ('111','T12');



S. exam\_subject Table

CREATE TABLE `exam\_subject` (

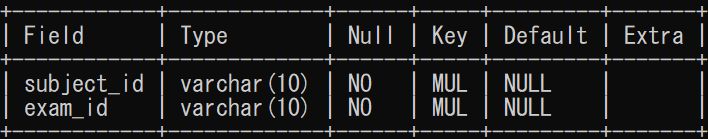
`subject\_id` VARCHAR(10) NOT NULL,

`exam\_id` VARCHAR(10) NOT NULL,

FOREIGN KEY (`subject\_id`) REFERENCES Subject(`subject\_id`),

FOREIGN KEY (`exam\_id`) REFERENCES Eximination(`exam\_id`)

);

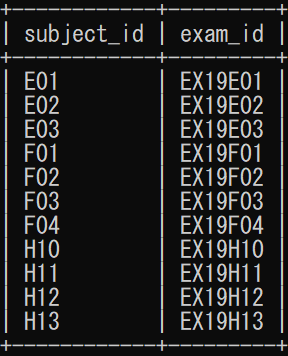


INSERT INTO `exam\_subject` VALUES

('E01','EX19E01'), ('E02','EX19E02'), ('E03','EX19E03'),

('F01','EX19F01'), ('F02','EX19F02'), ('F03','EX19F03'), ('F04','EX19F04'),

('H10','EX19H10'), ('H11','EX19H11'), ('H12','EX19H12'), ('H13','EX19H13');



T. student\_course Table

CREATE TABLE `student\_course` (

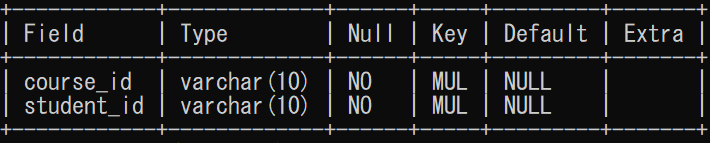
`course\_id` VARCHAR(10) NOT NULL,

`student\_id` VARCHAR(10) NOT NULL,

FOREIGN KEY (`course\_id`) REFERENCES Courses(`course\_id`),

FOREIGN KEY (`student\_id`) REFERENCES Student(`student\_id`)

);

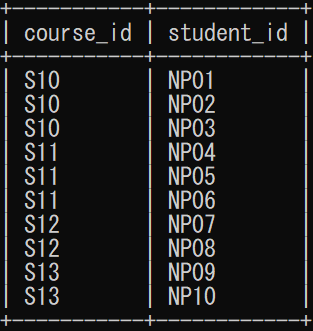


INSERT INTO `student\_course` VALUES

('S10','NP01'), ('S10','NP02'), ('S10','NP03'),

('S11','NP04'), ('S11','NP05'), ('S11','NP06'),

('S12','NP07'), ('S12','NP08'), ('S13','NP09'), ('S13','NP10');



# Select Statements using Different Functions

1. Write a query to display total percentage of all student along with their student id.

a. Solution

SELECT

`student\_id` AS `Student ID`,

SUM(`sub\_marks`)/(COUNT(`subject\_id`)\*100)\*100 AS `Percentage`

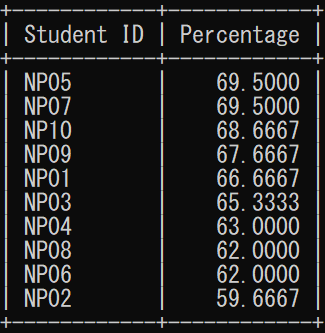
FROM `Marks`

GROUP BY `student\_id`

ORDER BY `Percentage`

DESC;

b. Result



2. Write a query to display total number of subject in each courses

a. Solution

SELECT

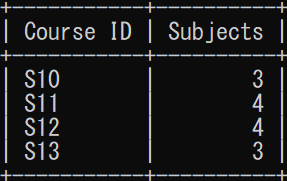
`course\_id` AS `Course ID`,

COUNT(\*) AS `Subjects`

FROM `Subject`

Group BY `course\_id`;

b. Result



3. Write a query to display highest marks obtained in subject E03 along with student id.

a. Solution

SELECT

`student\_id` AS `STUDENT`,

`sub\_marks` AS `Heighest Mark`,

`subject\_id` AS `Subject`

From `marks`

WHERE

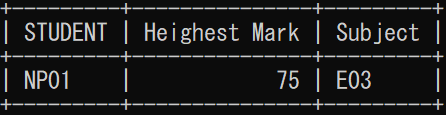
`sub\_marks` = (SELECT

MAX(`sub\_marks`)

FROM `Marks`

WHERE `subject\_id`= 'E03') AND `subject\_id`= 'E03';

b. Result



4. Write a query to display lowest marks obtained in subject T12 along with student id.

a. Solution

SELECT

`student\_id` AS `STUDENT`,

`sub\_marks` AS `Heighest Mark`,

`subject\_id` AS `Subject`

From `marks`

WHERE

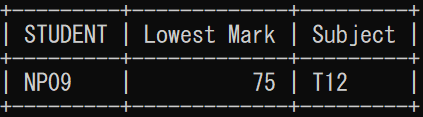
`sub\_marks` = (SELECT

MIN(`sub\_marks`)

FROM `Marks`

WHERE `subject\_id`= 'T12') AND `subject\_id`= 'T12';

b. Result



5. Write a query to display Date along with exam id which was held between '2019-05-9' AND '2019-05-10'.

a. Solution

SELECT

DATE(`exam\_date`) AS `EXAM DATE`,

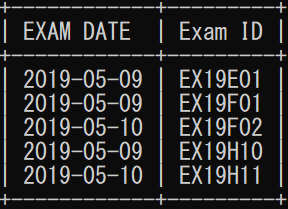
`exam\_id` AS `Exam ID`

FROM `eximination`

WHERE `exam\_date`

BETWEEN '2019-05-9' AND '2019-05-10';

b. Result



6. Write a query to display dates school had an exam to take in 2019.

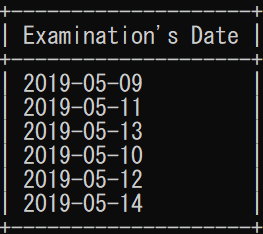
a. Solution

SELECT DISTINCT

`exam\_date` AS `Examination's Date`

from `eximination` WHERE `exam\_date` LIKE '2019%';

b. Result



7. Write a query to display the student id along with present(%) whose present percentage is greater than 80.

a. Solution

SELECT `student\_id` AS `Student ID`,

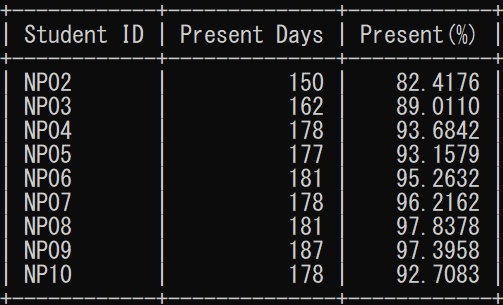
`present\_days` AS `Present Days`,

(`present\_days`/`total\_class`)\*100 AS `Present(%)`

FROM `attendance`

WHERE (`present\_days`/`total\_class`)\*100 > 80;

b. Result



8. Write a query to display the student is either good, average or poor in subject F04 and E01 respectively.

a. Solution

SELECT

`Student\_id` AS `Student ID`,

`subject\_id` AS `Subject ID`,

`sub\_marks` AS `Subject Marks`,

CASE

WHEN `sub\_marks` >= 70 THEN 'Good Student'

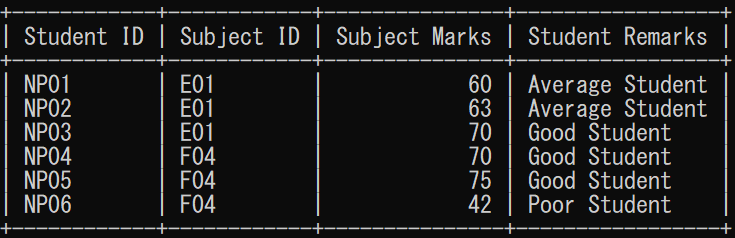
WHEN `sub\_marks` >= 60 THEN 'Average Student'

ELSE 'Poor Student'

END AS `Student Remarks`

FROM `Marks` WHERE `subject\_id` IN ('F04','E01');

b. Result



9. Write a query to display the total course marks of each student.

a. Solution

SELECT

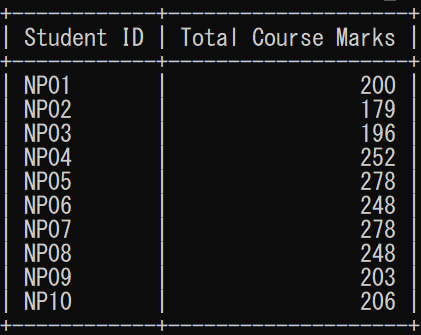
`student\_id` AS `Student ID`,

SUM(`sub\_marks`) AS `Total Course Marks`

FROM

`Marks` GROUP BY `student\_id`;

b. Result



10. Write a query to display teacher name in uppercase letter and in ascending order

a. Solution

SELECT

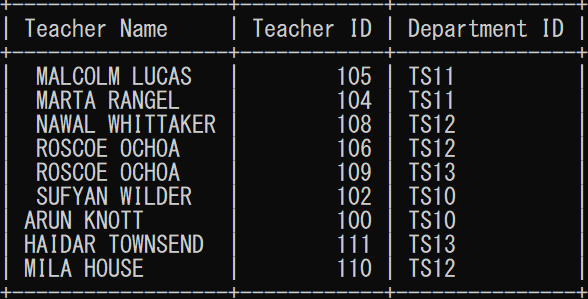
UCASE(`teacher\_name`) AS `Teacher Name`,

`teacher\_id` AS `Teacher ID`,

`department\_id` AS `Department ID`

FROM `Teacher` ORDER BY `teacher\_name` ASC;

b. Result



11. Write a query to trim and display teacher name in lowercase letter and in ascending order

a. Solution

SELECT

LCASE(`Teacher Name`),

`Teacher ID`,

`Department ID`

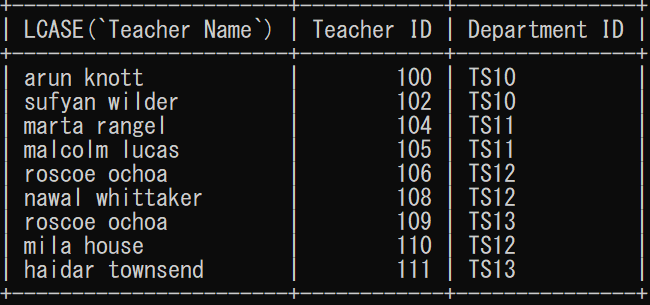
FROM (SELECT TRIM(`teacher\_name`) AS `Teacher Name`,

`teacher\_id` AS `Teacher ID`,

`department\_id` AS `Department ID`

FROM `Teacher` ORDER BY `teacher\_name` ASC) AS T;

b. Result



12. Write a query to replace first two letter of student id with ‘PP’ as new student ID

a. Solution

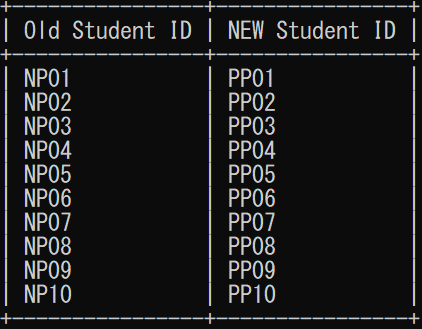
SELECT

`Student\_id` AS `Old Student ID`,

REPLACE(`student\_id`, 'NP', 'PP') AS `NEW Student ID`

FROM `Student`;

b. Result



13. Write a query to display present days over total date from table attendance of class LT02

a. Solution

SELECT

Student.student\_id AS `student ID`,

CONCAT(`present\_days`,' \/ ',`total\_class`) AS `Present/Total`,

`class\_id` AS `Class ID`

FROM

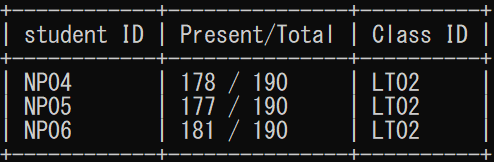
`student`, `attendance`

WHERE

student.student\_id = attendance.student\_id

AND Student.class\_id = 'LT02';

b. Result



14. Write a query to display the average marks of student having student id NP04

a. Solution

SELECT

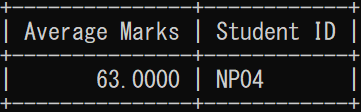
AVG(`sub\_marks`) AS `Average Marks`,

`student\_id` AS `Student ID`

FROM `marks`

WHERE `student\_id`='NP04';

b. Result



15. Write a query to count total number of books in library.

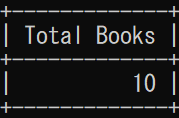
a. Solution

SELECT

COUNT(`book\_id`) AS `Total Books`

FROM `books`;

b. Result



16. Write a query to extract 3 letters from left of student name.

a. Solution

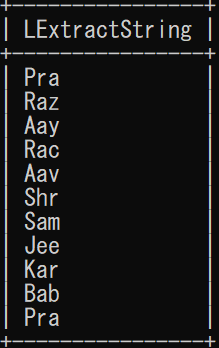
SELECT

LEFT(`student\_name`, 3)

AS `LExtractString`

FROM `student`;

b. Result



17. Write a query to extract 3 letters from right of student name.

a. Solution

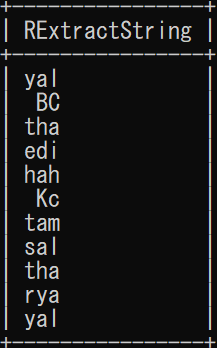
SELECT

RIGHT(`student\_name`, 3)

AS `RExtractString`

FROM `student`;

b. Result



18. Write a query to extract 3 letters from middle of student name along with their student id.

a. Solution

SELECT

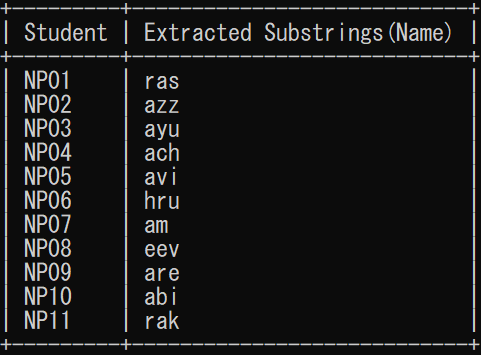
`student\_id` AS `Student`,

MID(Student\_name, 2, 3)

AS `Extracted Substrings(Name)`

FROM Student;

b. Result



19. Write a query to display name of book in top of books table when ordered by book id.

a. Solution

Select

`book\_name` AS `First Book in Table`

FROM

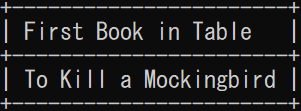
`Books`

ORDER BY `book\_id`

ASC

LIMIT 1;

b. Result



20. Write a query to display name of book in bottom of books table when ordered by book id.

a. Solution

Select

`book\_name` AS `Last Book in Table`

FROM

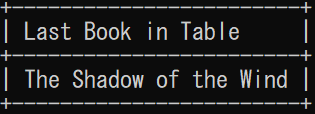
`Books`

ORDER BY `book\_id`

DESC

LIMIT 1;

b. Result



# Select Statements using Sub Query

1. Write a query to display total percentage and find the student with highest score among them.

a. Solution

SELECT

`student\_id` AS `Top Scorer`,

SUM(`sub\_marks`)/(COUNT(`subject\_id`)\*100)\*100 AS `Percentage`

FROM `Marks`

GROUP BY `student\_id`

HAVING (SUM(`sub\_marks`)/(COUNT(`subject\_id`)\*100)\*100) =

(

SELECT MAX(`Percentage`)

FROM (

SELECT

`student\_id` AS `Student ID`,

SUM(`sub\_marks`)/(COUNT(`subject\_id`)\*100)\*100 AS `Percentage`

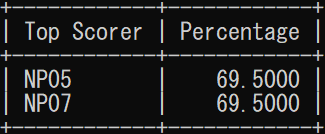
FROM `Marks`

GROUP BY `student\_id`

ORDER BY `Percentage` DESC) AS `MaxGrade`

);

b. Result



2. Write a query to display Student ID, Student name, Parent Name after joining two tables parent and student using sub query.

a. Solution

SELECT

`student\_id` AS `Student ID`,

`student\_name` AS `Student Name`,

`parent\_name` AS `Parent Name`

FROM (

SELECT

Student.student\_id,

Student.student\_name,

Parent.parent\_name,

Parent.phone\_number,

Student.class\_id

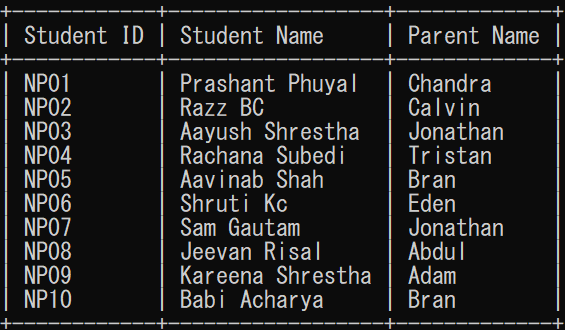
FROM `Student`

INNER JOIN `Parent`

ON Student.student\_id = Parent.student\_id

) AS JoinedTable;

b. Result



3. Write a query to display student ID, Parent Name, Phone Number, parent address to parent name starting with C using sub query.

a. Solution

SELECT

`student\_id` AS `Studuent ID`,

`parent\_name` AS `Parent Name`,

`phone\_number` AS `Parent ph. Number`,

`parent\_address` AS `Address`

FROM `Parent`

WHERE `parent\_name`

IN (

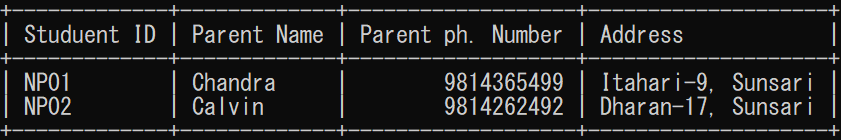
SELECT `parent\_name`

FROM `parent`

WHERE `parent\_name` LIKE 'C%'

);

b. Result



# Select Statements using Count and Group Functions

1. Write a query to display total percentage of all students.

a. Solution

SELECT

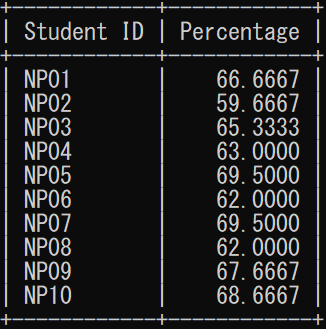
`student\_id` AS `Student ID`,

SUM(`sub\_marks`)/(COUNT(`subject\_id`)\*100)\*100 AS `Percentage`

FROM `Marks`

GROUP BY `student\_id`;

b. Result



2. Write a query to display all the departments along with the total number of teachers in each department.

a. Solution

SELECT

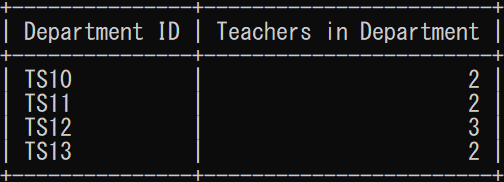
`department\_id` AS `Department ID`,

COUNT(\*) AS `Teachers in Department`

FROM `Teacher`

Group BY `department\_id`;

b. Result



# Select Statements using Different Joins

1. Write a query to join table’s student, Courses and class with attributes Student ID, Course ID, and Class ID Using INNER JOIN.

a. Solution

SELECT

S.student\_id `Student ID`,

C.course\_id `Course ID`,

Cl.class\_id `Class ID`

FROM

Student AS S

INNER JOIN

student\_course AS sc

ON S.student\_id = Sc.student\_id

INNER JOIN

Courses AS C

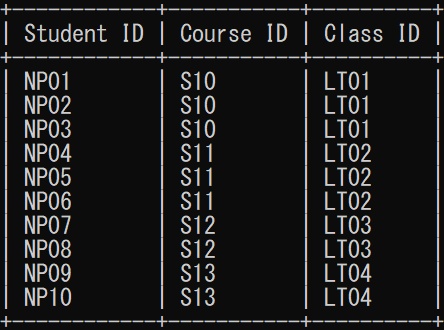
ON C.course\_id = Sc.course\_id

INNER JOIN

Class AS Cl

ON Cl.class\_id = S.class\_id;

b. Result



2. Write a query to join two tables Books and members with attributes Student ID, Member ID, Books ID and books name using LEFT JOIN.

a. Solution

SELECT

B.book\_id `Book ID`,

B.book\_name `Book Name`,

M.member\_id `Member ID`,

M.student\_id `Student ID`

FROM

Books AS B

LEFT JOIN

issued\_books AS I

ON B.book\_id = I.book\_id

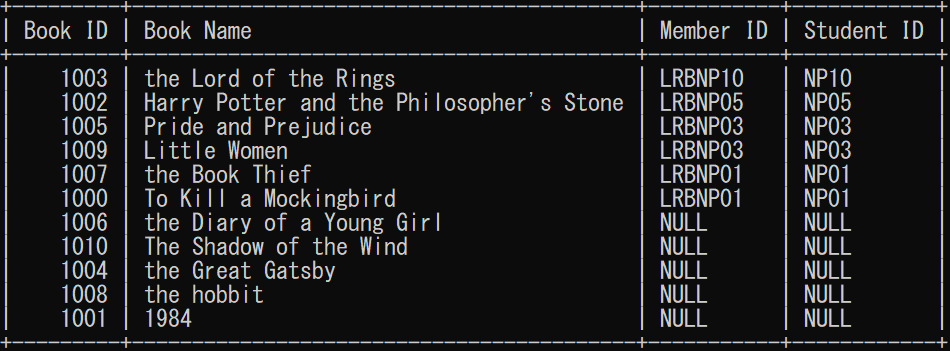
LEFT JOIN

Members AS M

ON

M.member\_id = I.member\_id ORDER BY M.member\_id DESC;

b. Result



3. Write a query to join tables Books and members with attributes Student ID, Member ID, Books ID and books name using RIGHT JOIN.

a. Solution

SELECT

B.book\_id `Book ID`,

B.book\_name `Book Name`,

M.member\_id `Member ID`,

M.student\_id `Student ID`

FROM

Books AS B

RIGHT JOIN

issued\_books AS I

ON B.book\_id = I.book\_id

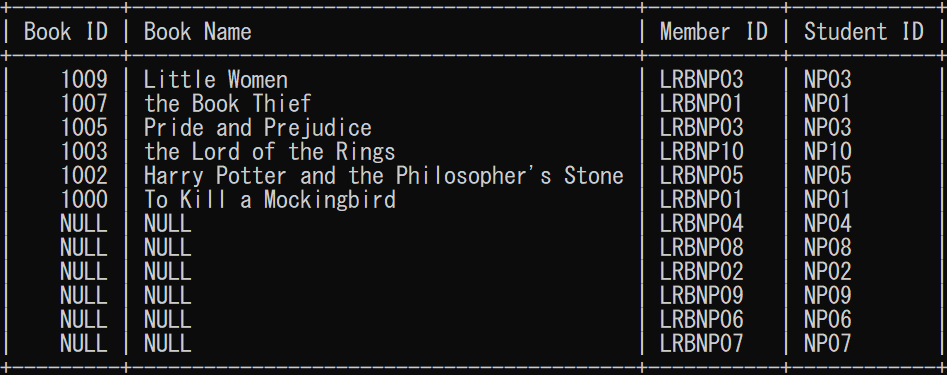
RIGHT JOIN

Members AS M

ON

M.member\_id = I.member\_id ORDER BY B.book\_id DESC;

b. Result



4. Write a query to join tables Department, Teachers, Subject and Period Hour with attributes Teacher ID, Teacher Name, Department ID, Subject ID and Time period using JOIN.

a. Solution

SELECT

T.teacher\_id AS `Teacher ID`,

T.teacher\_name `Teacher Name`,

D.department\_id `Department ID`,

S.subject\_id `Subejct ID`,

PH.time\_period `Time`

FROM

Department AS D

JOIN

Teacher AS T

ON D.department\_id = T.department\_id

JOIN

teacher\_teaches AS TT

ON T.teacher\_id = TT.teacher\_id

JOIN

Subject AS S

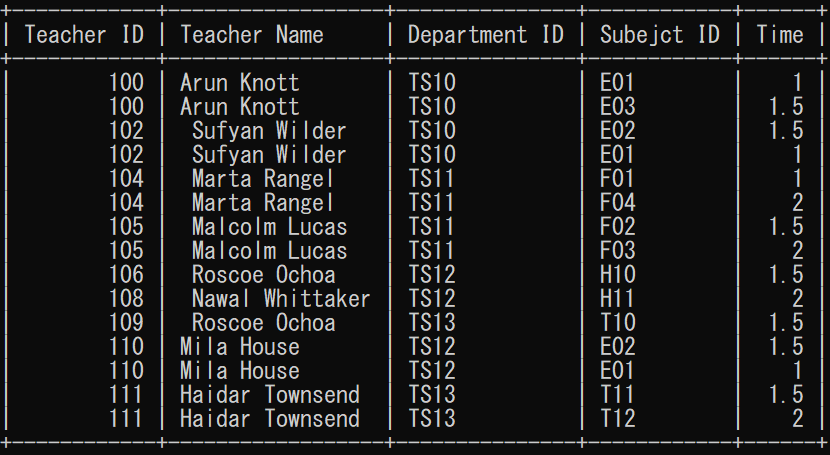
ON S.subject\_id = TT.subject\_id

JOIN

PeriodHour AS PH

ON S.subject\_id = PH.subject\_id;

b. Result



5. Write a query to join tables Student, Eximination, Subject and class with attributes Exam ID, Student ID and Subject ID without using JOIN.

a. Solution

SELECT

E.exam\_id `Exam ID`,

S.student\_id `Student ID`,

Su.subject\_id `Subject ID`

FROM

Student AS S,

Eximination AS E,

Subject AS Su,

Class AS Cl,

exam\_subject AS Es

WHERE

S.class\_id = Cl.class\_id

AND

Cl.class\_id = Su.class\_id

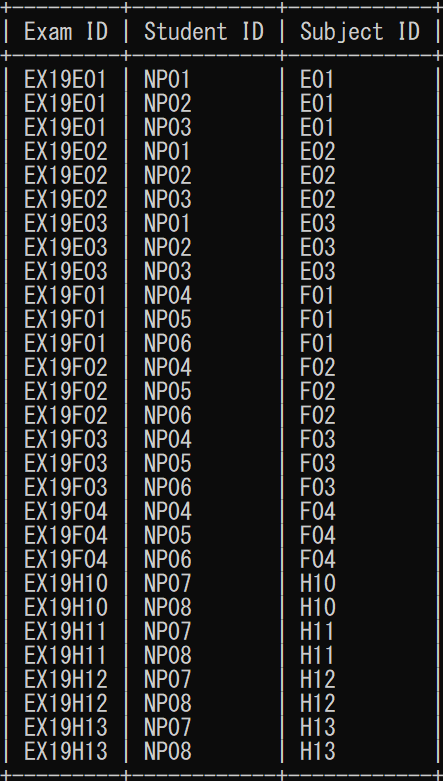
AND

Su.subject\_id = Es.subject\_id

AND

E.exam\_id = Es.exam\_id;

b. Result



# Insert Statement

1. Write a query to insert student id, student name, class id, roll no, DOB and gender of student table.

a. Solution

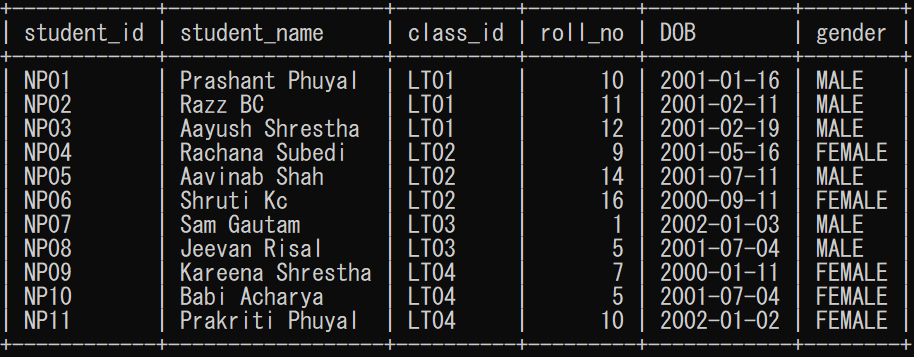
INSERT INTO `Student` ( `student\_id`, `student\_name`, `class\_id`, `roll\_no`, `DOB`, `gender`)

VALUES (

'NP11','Prakriti Phuyal','LT04','10','2002-01-02','FEMALE'

);

b. Result



2. Write a query to insert parent name, student id, phone number, parent address of parent table.

a. Solution

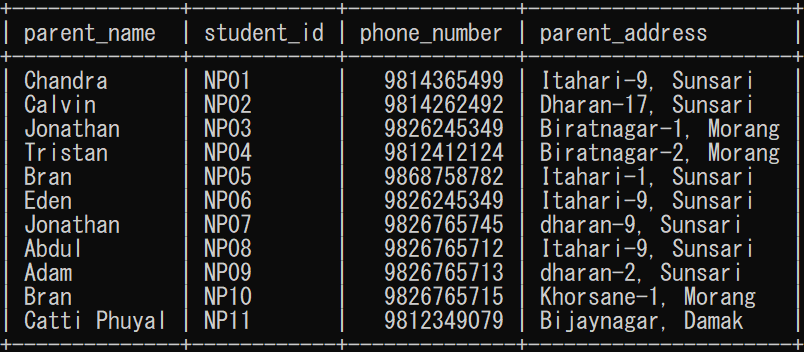
INSERT INTO `Parent` ( `parent\_name`, `student\_id`, `phone\_number`, `parent\_address`)

VALUES (

'Catti Phuyal','NP11','9812349079','Bijaynagar, Damak'

);

b. Result



3. Write a query to add student NP11 in enrolling course.

a. Solution

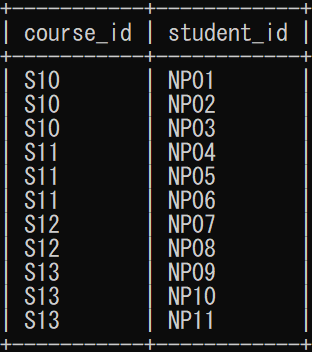
INSERT INTO `student\_course` (`course\_id`,`student\_id`)

VALUES (

'S13', 'NP11'

);

b. Result



4. Write a query to add student NP11 in members table as a member of library.

a. Solution

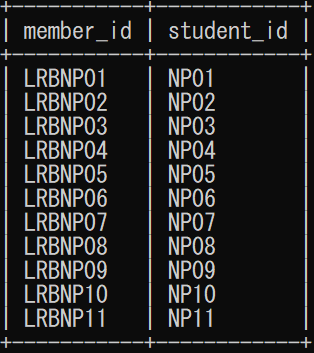
INSERT INTO `Members` (`member\_id`,`student\_id`)

VALUES (

'LRBNP11', 'NP11'

);

b. Result



5. Write a query to insert books in parent table.

a. Solution

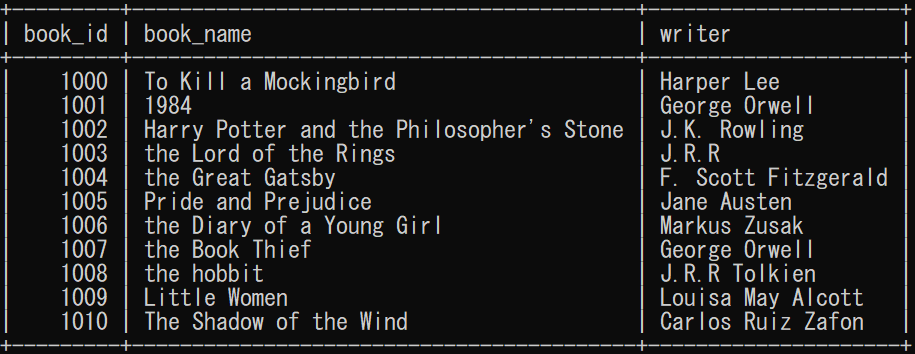
INSERT INTO `Books` (`book\_name`,`writer`)

VALUES (

'The Shadow of the Wind', 'Carlos Ruiz Zafon'

);

b. Result



# Update Statement

1. Write a query to update student name, class id, gender from student table of student NP07.

a. Solution

UPDATE `Student`

SET

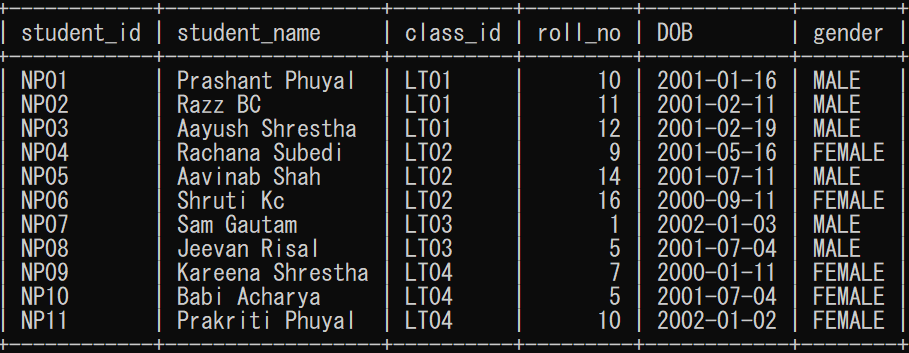
`student\_name` = 'Samyog Gotam',

`class\_id` = 'LT04',

`gender` = 'FEMALE'

WHERE `student\_id` = 'NP07';

b. Result



2. Write a query to update student staff name, work role, staff phone number from staff table of staff id SF06.

a. Solution

UPDATE `Staff`

SET

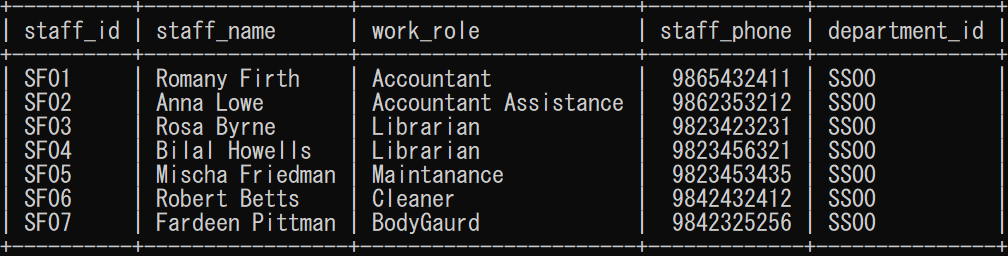
`staff\_name` = 'Rose Khadka',

`work\_role` = 'Event Manager',

`staff\_phone` = '9842223333'

WHERE `staff\_id` = 'SF06';

b. Result



3. Write a query to update teacher address and teacher department from teacher table of teacher having id 110

a. Solution

UPDATE `Teacher`

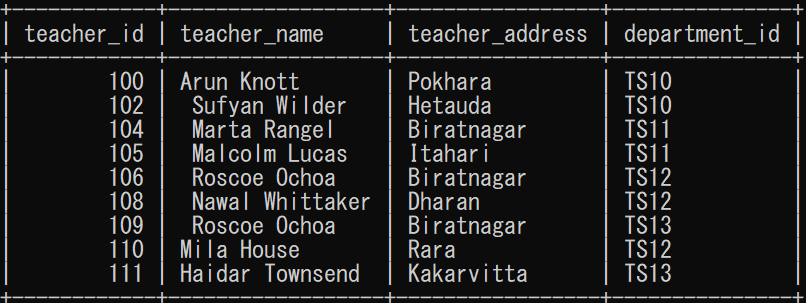
SET

`teacher\_address` = 'Rara Lake',

`department\_id` = 'TS10'

WHERE `teacher\_id` = 110;

b. Result



4. Write a query to update teacher subject with id 110 has to teach from table teacher\_teaches.

a. Solution

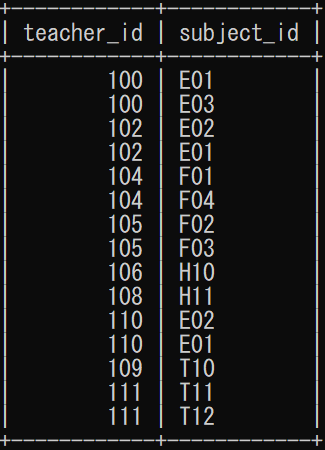
UPDATE `teacher\_teaches`

SET

`subject\_id` = 'E02'

WHERE teacher\_id = '110' Limit 1;

b. Result



5. Write a query to update time period of subject H13 from table periodHour.

a. Solution

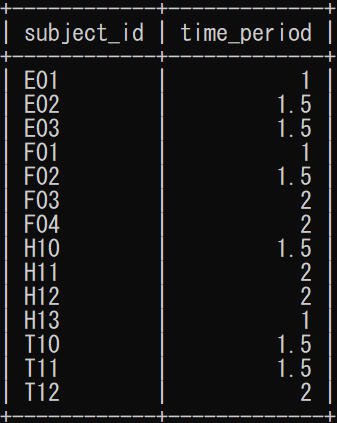
UPDATE `periodhour`

SET

`time\_period` = 1

WHERE subject\_id = 'H13';

b. Result



# Delete Statement

1. Write a query to remove student NP11 from course.

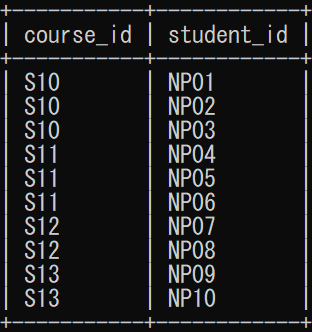
a. Solution

DELETE

FROM `student\_course`

WHERE `student\_id`= 'NP11';

b. Result



2. Write a query to remove student NP11 library membership.

a. Solution

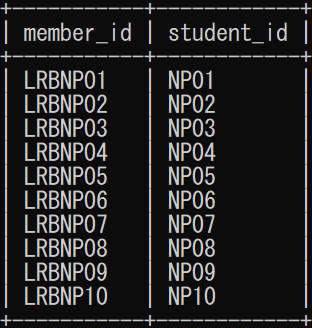
DELETE

FROM

`members`

WHERE `student\_id`= 'NP11';

b. Result



3. Write a query to delete parent data of student NP11 from parent table.

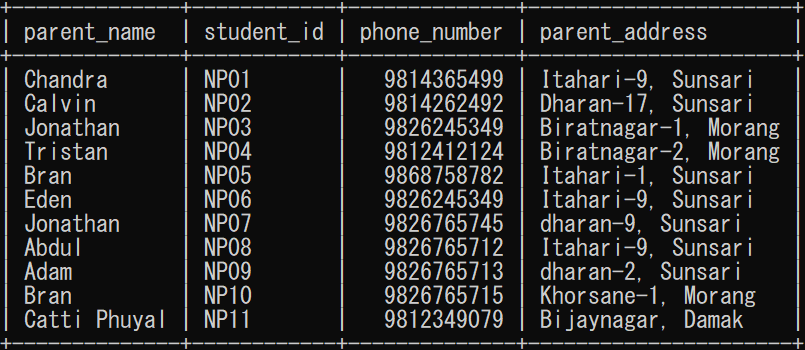
a. Solution

DELETE

FROM `Parent`

WHERE `student\_id`= 'NP11';

b. Result



4. Write a query to delete student data of student NP11 from student table.

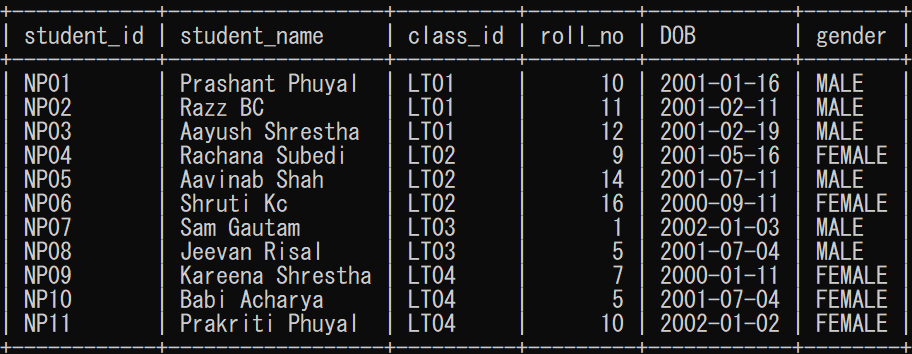
a. Solution

DELETE

FROM `Student`

WHERE `student\_id`= 'NP11';

b. Result



5. Write a query to delete issued record of book having id 1006 from issued\_book table.

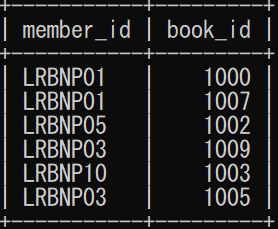
a. Solution

DELETE

FROM `issued\_books`

WHERE `book\_id`= 1006

b. Result



# Normalization

Database Normalization is the process of organizing data in the database. It is the systematic process of eliminating data redundancy and undesirable characteristics like Update, Insertion and Deletion Anomalies. If the data is not normalized and have data repetition then it would occupy more space and makes difficult in handling operations like deletion and insertion. The added advantage of getting an organized data enhances the performance level. The main purpose of normalization is:

Increased consistency: Database normalization decreases the repetition of data placing data in one place and one place only, reducing the possibility of inconsistent data.

Easier object-to-data mapping: Database maintained under normalization rules in general are conceptually closer to object-oriented schemas as the object-oriented achievements of enhancing high cohesion and loose coupling between classes results in similar solutions.

Normalization rules are divided into following normal forms:

* First Normal Form (1NF)
* Second Normal Form (2NF)
* Third Normal From (3NF)

## First Normal Form

The first normal form expects to follow a few simple rules while designing the database:

* Each column should contain atomic values.
* A column should contain values that are of the same type.
* Each column should have a unique name.
* Order in which data is saved doesn’t matter.

1NF Example

Let us create one Student table with Student Name, Roll No and Subject as shown below:

|  |  |  |
| --- | --- | --- |
| **Rollno** | **name** | **Subject** |
| 101 | Prashant | OS, CN |
| 103 | Aayush | JAVA |
| 102 | Aavinab | C, C++ |

Here out of the 3 student 2 has more than 1 subjects, and I have stored in them in single column but as per first normal form each table should contain atomic values, which leads to the violation of 1NF.

So, we can solve this problem by breaking the 2 values in single column to atomic values as shown below:

|  |  |  |
| --- | --- | --- |
| **Rollno** | **name** | **Subject** |
| 101 | Prashant | OS |
| 101 | Prashant | CN |
| 103 | Aayush | JAVA |
| 102 | Aavinab | C |
| 102 | Aavinab | C++ |

(Study Tonight, 2020)

Although there is a repeatation of data in table, but values for subject column is atomic for each row.

## Second Normal Form

For a table to be in second normal form we should follow a few simple rules:

* It should be in 1st normal form
* It should not have any partial dependencies

Now, let us create a score table with attribute score id, student id, subject id, marks and teacher.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Score\_id** | **Student\_id** | **Subject\_id** | **marks** | **Teacher** |
| 1 | 10 | 1 | 82 | Mr. OS |
| 2 | 10 | 2 | 77 | Mrs. CN |
| 3 | 11 | 1 | 85 | Mr. JAVA |
| 4 | 11 | 2 | 82 | Mr. C |
| 5 | 11 | 4 | 95 | Mr. C++ |

The table follows 1NF rules as each attribute has atomic values. However, it does not follows second normal form because teacher column depends on subject id but it has nothing to do with student which leads to partial dependency and violates the second normal form.

There are many different solution to remove partial dependency the only objective is to remove teacher form score table.

One of the way is to move the teacher column to the subject table where it fits appropriately along with the subject names.

|  |  |  |
| --- | --- | --- |
| **Subject\_id** | **Subject name** | **Teacher** |
| 1 | OS | Mr. OS |
| 2 | CN | Mrs. CN |
| 1 | JAVA | Mr. JAVA |
| 2 | C | Mr. C |
| 4 | C++ | Mr. C++ |

Also, we can create another separate table for the teacher and use the teacher’s id where ever we want.

|  |  |
| --- | --- |
| **Subject\_id** | **Teacher** |
| 1 | Mr. OS |
| 2 | Mrs. CN |
| 3 | Mr. JAVA |
| 4 | Mr. C |
| 5 | Mr. C++ |

(Study Tonight, 2020)

## Third Normal Form

For a table to be in third normal form we should follow a few simple rules:

* It should be in 2st normal form
* It should not have transitive dependencies

Now, let us create a score table with attribute score id, student\_id, subject\_id, marks and exam\_name and total\_marks.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Score\_id** | **Student\_id** | **Subject\_id** | **marks** | **Exam\_name** | **Total\_marks** |
| 101 | 10 | 1 | 82 | OS | 120 |
| 102 | 10 | 2 | 77 | CN | 130 |
| 111 | 11 | 1 | 85 | JAVA | 110 |
| 112 | 11 | 2 | 82 | C | 140 |
| 114 | 11 | 4 | 95 | C++ | 120 |

Here, score\_id is formed with the combination of two keys student\_id and subject\_id as a composite key. Hence, primary id is both student\_id and subject\_id and all other attributes depends on it expect total marks which depends on exam\_name but exam name is not a part of primary key which leads to transitive dependency(when attribute on table depends on some non-prime attributes) which violates the third normal form.

Solution to this problem is we should take exam\_name and total\_marks and put them on exam table and use exam id where ever it requires.

|  |  |  |
| --- | --- | --- |
| **Exam\_id** | **Exam\_name** | **Total\_marks** |
| 1 | OS | 120 |
| 2 | CN | 130 |
| 3 | JAVA | 110 |
| 4 | C | 140 |
| 5 | C++ | 120 |

In this way, score table is in third normal form.

(Study Tonight, 2020)

# Conclusion

This project engaged me in understanding about database and SQL queries. The report contents successfully accomplished its achievement by allowing me to work with Xampp, MySQL Workbench and testing and learning SQL queries, which in turn helped me to understand database management system and further more. This coursework helped me to become more familiar with the concepts of database management, about its entities, attributes, ER diagram and queries. Syntax and functions to create, update and insert attributes were also a best experience. I felt Xampp as a complex shell for writing SQL queries so I used MySQL workbench first and after it I exported the script file from workbench and imported on xampp for further report screenshot purpose.

The coursework was an interesting experience that helped me learn the ways of database management systems and to operate them. I enjoyed a lot while doing this work although many problems were encountered throughout the coursework along with its process but they were quickly solved by the help respected teachers. I want to thank Mr. sachit tandukar sir, Mr. Subiran sir and Mr. Deepson sir for guide and support throughout the coursework. (w3schools.com, n.d.) (codeacademy, n.d.) (study tonight, n.d.)

# References

codeacademy. (n.d.). *List of SQL Commands*. Retrieved from code academy: https://www.codecademy.com/articles/sql-commands

Study Tonight. (2020). *studytonight*. Retrieved from Normalization of Database: https://www.studytonight.com/dbms/database-normalization.php

study tonight. (n.d.). *What are SQL Functions?* Retrieved from studytonight: https://www.studytonight.com/dbms/sql-function.php

w3schools.com. (n.d.). *mysql functions*. Retrieved from w3schools: https://www.w3schools.com/sql/sql\_ref\_mysql.asp