Coaching Institute Database Management system

Educational institutions need to manage huge data of students. To streamline the process of managing student data, a lot of educational institutions are investing in a high-performance student database management system.

Problem statement:

This coaching Institute has one training center.It is managed by one/more Director's/Admin panel. This canter has three Admin who manages the work of institute, instructors,courses and students. Institute have one or more courses. Student can choose one or many courses. Each course has one instructor. Each instructor can taught many students and each student can learn from many instructors. Each student can only receive overall one grade for a course, but grades can belong to more than one student. This database consist of Admin,instructors, course, students, and grades.

Solution:

❒ **Identify Entity and Members**:

This coaching Institute has one training center.It is managed by one/more Director's/**Admin** panel. This canter has three Admin who manages the work of institute, instructors,courses and students. Institute have one or more **courses**. **Student** can choose one or many courses. Each course has one instructor. Each **instructor** can taught many students and each student can learn from many instructors. Each student can only receive overall one **grade** for a course, but grades can belong to more than one student.

This database consist of **Admin, instructors, course, students** as entities.

❒ **Decide Relationships, Cardinality:**

In This Institute there are three admins who manages the work of institute, instructors, courses and students

- **One** Institute have **one or more** courses
- **One or many** Student can choose **one or many** courses
- **One** course has **one** instructor
- **One** instructor can taught **many** students and **one** student can learn from **many** instructors
- **One** Course has **one** fee
- **One** student can only receive overall **one** grade for a course, but **one** grades can belong to **many** student

❒ **Draw Entities :**

- Admin-instructor: One or many Admins manages one or more instructors
- Admim-students:One or many Admins manages and keep details of one or many students
- Admin-Course: One or many Admins manages and keep details of one or many Courses
- Student-Course : one or many Students must have one or many courses

- Student-grade: one student have one grade for one course
- Course-Student : One course can be offered to many students
- Course-Instructor : One course can be taught by one Instructor
- Course-fees: One course has one fixed fee
- instructor-Course : One Instructor will teach one course

❒ Draw Attribute Separately:

- Admin can manage number of instructor, courses and student with Admin number and Admin name
- Student have Student Number,Name,address ,phone Number ,Course Number and grade
- Instructor have Number,Name,address and Course Number
- Course have Number, Name and fees

❒ Normalization in DBMS:

Normalization is the branch of relational theory that provides design insights. It is the process of determining how much redundancy exists in a table.

The goals of normalization are to:

- Be able to characterize the level of redundancy in a relational schema
- Provide mechanisms for transforming schemas in order to remove redundancy

Means Normalization is Process of removing redundancy , data duplication and elimination of anomalies and preventing loss of information.

- **Without Normalization:**

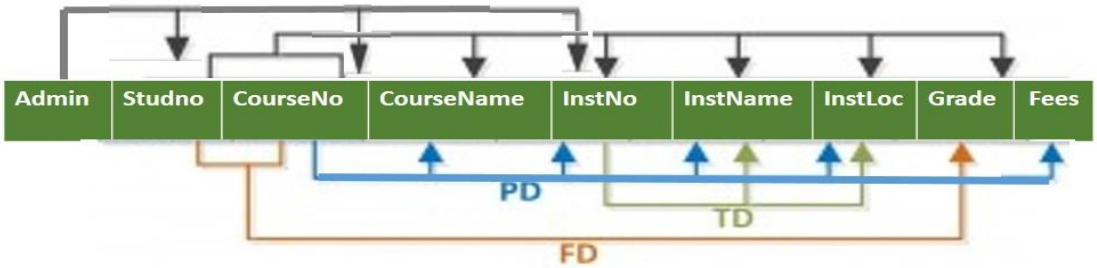
Coaching Institute Database Management system

Without Normalization

Admin Table

Admin_No	Admin_Name	StudentNo	StudentName	Address	Contact	CourseNo	CourseName	InstructorNo	InstructorName	InstructorLocation	Grade	Fees
1	Vinayak Ingale	101	Priya Kasalkar	Thane	9896857485	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A	68000
						C202	Python Development	IN102	Vipul Kumar	Airoli	A	72000
		102	Shravani Raut	Kalva	9945125847						A+	
		103	Sagar Naik	Diva	7785451236	C201	Web Development	IN101	Tejraj Singh Hada	Thane	B	68000
						C203	Android Development	IN103	Roshni Sharma	Kalyan	A+	87000
		104	Abhishek Kulkarni	Ghansoli	9632145678						A	
		105	Diksha Mane	Dombivali	9863524174	C201	Web Development	IN101	Tejraj Singh Hada	Thane	B	68000
						C202	Python Development	IN102	Vipul Kumar	Airoli	A	72000
		106	Sudhir Gurav	Koper	9456321584						A+	
2	Anjali Deshmukh	107	Ranjita Desai	Thane	9485632158	C202	Python Development	IN102	Vipul Kumar	Airoli	B	72000
						C201	Web Development	IN101	Tejraj Singh Hada	Thane	A	68000
		108	Shweta Rasal	Dombivali	9856743284						B+	
		109	Minakshi Patil	Diva	9743186392						C	
		110	Rasika Deshmukh	Rabale	9482369712	C202	Python Development	IN102	Vipul Kumar	Airoli	A+	72000
						C203	Android Development	IN103	Roshni Sharma	Kalyan	B	87000
3	Rutuja Pednekar	111	Vijaya Dhanavade	Airoli	7758369853	C203	Android Development	IN103	Roshni Sharma	Kalyan	A	87000
		112	Arati sarnaik	Ghatkoper	7709220540						A	
		113	Seema Uchale	Mulund	7752148250						B	
		114	Geeta Patade	Thane	9586241854						B+	
		115	Gouresh Munagek	Diva	9640250120						A	

❖ **Dependencies in Coaching Institute system:**



❖ First Normal Form (1NF):

In the first normal form, only single values are permitted at the intersection of each row and column; hence, there are no repeating groups.

To normalize a relation that contains a repeating group, remove the repeating group and form two new relations.

The Primary Key of the new relation is a combination of the PK of the original relation plus an attribute from the newly created relation for unique identification

<u>Process for 1NF:</u>

Here in this database I use the **Admin** table below, from a Coaching Institute database.

Admin(Admin_no,Admin_Name,StudentNo, StudentName,Address,contact, CourseNo,

CourseName, InstructorNo, InstructorName, InstructorLocation, Grade, Fees)

In the Admin table, the repeating group is the course information. A student can take many courses and Admin manages instructor,course and students.

1) Remove the repeating group. In this case, it's the course information for each student and Admin manages student, instructor and course.

2) Identify the PK for new table

The PK must uniquely identify the attribute value (AdminNo, StudentNo and CourseNo).After removing all the attributes related to the Admin, course and student, left with the student course table (StudentEnCourse) and student table. The Admin table is now in first normal form with the repeating group removed.

1)The Student table (Student) is now in first normal form with the repeating group removed With Admin(AdminNo,Admin_Name),Student (StudentNo, StudentName, Address,contact),StudentCourse (StudentNo, CourseNo, CourseName,InstructorNo, InstructorName, InstructorLocation, Grade,Fees)

First normal form

Admin Table — Remove the repeating group and Identify the PK

Admin_No	Admin_Name	StudentNo	StudentName	Address	Contact	CourseNo	CourseName	InstructorNo	InstructorName	InstructorLocation	Grade	Fees
1	Vinayak Ingale	101	Priya Kasalkar	Thane	9896857485	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A	68000
1	Vinayak Ingale	102	Shravani Raut	Kalva	9945125847	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A+	68000
1	Vinayak Ingale	103	Sagar Naik	Diva	7785451236	C201	Web Development	IN101	Tejraj Singh Hada	Thane	B	68000
1	Vinayak Ingale	104	Abhishek Kulkarni	Ghansoli	9632145678	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A	68000
1	Vinayak Ingale	105	Diksha Mane	Dombivali	9863524174	C201	Web Development	IN101	Tejraj Singh Hada	Thane	B	68000
1	Vinayak Ingale	106	Sudhir Gurav	Koper	9456321584	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A+	68000
2	Anjali Deshmukh	107	Ranjita Desai	Thane	9485632158	C202	Python Development	IN102	Vipul Kumar	Airoli	B	72000
2	Anjali Deshmukh	108	Shweta Rasal	Dombivali	9856743284	C202	Python Development	IN102	Vipul Kumar	Airoli	B+	72000
2	Anjali Deshmukh	109	Minakshi Patil	Diva	9743186392	C202	Python Development	IN102	Vipul Kumar	Airoli	C	72000
2	Anjali Deshmukh	110	Rasika Deshmukh	Rabale	9482369712	C202	Python Development	IN102	Vipul Kumar	Airoli	A+	72000
3	Rutuja Pednekar	111	Vijaya Dhanavade	Airoli	7758369853	C203	Android Development	IN103	Roshni Sharma	Kalyan	A	87000
3	Rutuja Pednekar	112	Arati sarnaik	Ghatkoper	7709220540	C203	Android Development	IN103	Roshni Sharma	Kalyan	A	87000
3	Rutuja Pednekar	113	Seema Uchale	Mulund	7752148250	C203	Android Development	IN103	Roshni Sharma	Kalyan	B	87000
3	Rutuja Pednekar	114	Geeta Patade	Thane	9586241854	C203	Android Development	IN103	Roshni Sharma	Kalyan	B+	87000
3	Rutuja Pednekar	115	Gouresh Munagek	Diva	9640250120	C203	Android Development	IN103	Roshni Sharma	Kalyan	A	87000
1	Vinayak Ingale	101	Priya Kasalkar	Thane	9896857485	C202	Python Development	IN102	Vipul Kumar	Airoli	A	72000
1	Vinayak Ingale	103	Sagar Naik	Diva	7785451236	C203	Android Development	IN103	Roshni Sharma	Kalyan	A+	87000
1	Vinayak Ingale	105	Diksha Mane	Dombivali	9863524174	C202	Python Development	IN102	Vipul Kumar	Airoli	A	72000
2	Anjali Deshmukh	107	Ranjita Desai	Thane	9485632158	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A	68000
2	Anjali Deshmukh	110	Rasika Deshmukh	Rabale	9482369712	C203	Android Development	IN103	Roshni Sharma	Kalyan	B	87000

2)Identify Primary key:

First normal form

Remove the repeating group and

Admin

Admin_No	Admin_Name
1	Vinayak Ingale
2	Anjali Deshmukh
3	Rutuja Pednekar

Admin_manage

Admin_No	StudentNo	CourseNo	InstructorNo
1	101	C201	IN101
1	102	C201	IN101
1	103	C201	IN101
1	104	C201	IN101
1	105	C201	IN101
1	106	C201	IN101
2	107	C202	IN102
2	108	C202	IN102
2	109	C202	IN102
2	110	C202	IN102
3	111	C203	IN103
3	112	C203	IN103
3	113	C203	IN103
3	114	C203	IN103
3	115	C203	IN103
1	101	C202	IN102
1	103	C203	IN103
1	105	C202	IN102
2	107	C201	IN101
2	110	C203	IN103

Student

StudentNo	StudentName	Address	Contact
101	Priya Kasalkar	Thane	9896857485
102	Shravani Raut	Kalva	9945125847
103	Sagar Naik	Diva	7785451236
104	Abhishek Kulkarni	Ghansoli	9632145678
105	Diksha Mane	Dombivali	9863524174
106	Sudhir Gurav	Koper	9456321584
107	Ranjita Desai	Thane	9485632158
108	Shweta Rasal	Dombivali	9856743284
109	Minakshi Patil	Diva	9743186392
110	Rasika Deshmukh	Rabale	9482369712
111	Vijaya Dhanavade	Airoli	7758369853
112	Arati sarnaik	Ghatkoper	7709220540
113	Seema Uchale	Mulund	7752148250
114	Geeta Patade	Thane	9586241854
115	Gouresh Munagekar	Diva	9640250120

Cont.

n

Identify the PK

StudentCourse							
StudentNo	**CourseNo**	**CourseName**	**InstructorNo**	**InstructorName**	**InstructorLocation**	**Grade**	**Fees**
101	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A	68000
102	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A+	68000
103	C201	Web Development	IN101	Tejraj Singh Hada	Thane	B	68000
104	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A	68000
105	C201	Web Development	IN101	Tejraj Singh Hada	Thane	B	68000
106	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A+	68000
107	C202	Python Development	IN102	Vipul Kumar	Airoli	B	72000
108	C202	Python Development	IN102	Vipul Kumar	Airoli	B+	72000
109	C202	Python Development	IN102	Vipul Kumar	Airoli	C	72000
110	C202	Python Development	IN102	Vipul Kumar	Airoli	A+	72000
111	C203	Android Development	IN103	Roshni Sharma	Kalyan	A	87000
112	C203	Android Development	IN103	Roshni Sharma	Kalyan	A	87000
113	C203	Android Development	IN103	Roshni Sharma	Kalyan	B	87000
114	C203	Android Development	IN103	Roshni Sharma	Kalyan	B+	87000
115	C203	Android Development	IN103	Roshni Sharma	Kalyan	A	87000
101	C202	Python Development	IN102	Vipul Kumar	Airoli	A	72000
103	C203	Android Development	IN103	Roshni Sharma	Kalyan	A+	87000
105	C202	Python Development	IN102	Vipul Kumar	Airoli	A	72000
107	C201	Web Development	IN101	Tejraj Singh Hada	Thane	A	68000
110	C203	Android Development	IN103	Roshni Sharma	Kalyan	B	87000

- **How to update 1NF anomalies:**

StudentCourse (Student_No, CourseNo, CourseName, fees, InstructorNo, InstructorName, InstructorLocation, Grade,Fees)

Insertion Anomaly:

To add a new course, we need a student.

Update Anomaly:

When course information needs to be updated, we may have inconsistencies. if one student changes his/her address then we would have to update the table. But the problem is, if the table is not normalized one student can have multiple entries and while updating all of those entries one of them might get missed.

Deletion Anomaly:

To delete a student, we might also delete critical information about a course.

❖ Second Normal Form (2NF):

For the second normal form, the relation must first be in 1NF. The relation is automatically in 2NF if, and only if, the PK comprises a single attribute.

If the relation has a composite PK, then each non-key attribute must be fully dependent on the entire PK and not on a subset of the PK (i.e., there must be no partial dependency).

Process for 2NF:

To move to 2NF, a table must first be in 1NF.

The Student table is already in 2NF because it has a single-column PK.

When examining the Student Course table, we see that not all the attributes are fully dependent on the PK; specifically, all course information. The only attribute that is fully dependent is grade.

Identify the new table that contains the course information.

Identify the PK for the new table.

There are some new tables are shown below.

Admin(AdminNo,Admin_Name)

Admin_course(Admin_No,CourseNo)

Admin_Instr(Admin_No,InstructorNo)

Admin_stud(Admin_No,StudentNo)

Student (StudentNo, StudentName,Address,Contact)

StudenEncourse(StudentNo, CourseNo, Grade)

Coursedetails(CourseNo,CourseName, InstructorNo, InstructorName, InstructorLocation,fees)

Second normal form

It is in 1NF and all non-key attributes are fully functional dependent on the primary key

Admin

Admin_No	Admin_Name
1	Vinayak Ingale
2	Anjali Deshmukh
3	Rutuja Pednekar

Admin_course

Admin_No	CourseNo
1	C201
1	C202
1	C203
2	C201
2	C202
2	C203
3	C201
3	C202
3	C203

Admin_Stud

Admin_No	StudentNo
1	101
1	102
1	103
1	104
1	105
1	106
2	107
2	108
2	109
2	110
3	111
3	112
3	113
3	114
3	115

Admin_Instr

Admin_No	InstructorNo
1	IN101
1	IN102
1	IN103
2	IN101
2	IN102
2	IN103
3	IN101
3	IN102
3	IN103

Student

StudentNo	StudentName	Address	Contact
101	Priya Kasalkar	Thane	9896857485
102	Shravani Raut	Kalva	9945125847
103	Sagar Naik	Diva	7785451236
104	Abhishek Kulkarni	Ghansoli	9632145678
105	Diksha Mane	Dombivali	9863524174
106	Sudhir Gurav	Koper	9456321584
107	Ranjita Desai	Thane	9485632158
108	Shweta Rasal	Dombivali	9856743284
109	Minakshi Patil	Diva	9743186392
110	Rasika Deshmukh	Rabale	9482369712
111	Vijaya Dhanavade	Airoli	7758369853
112	Arati sarnaik	Ghatkoper	7709220540
113	Seema Uchale	Mulund	7752148250
114	Geeta Patade	Thane	9586241854
115	Gouresh Munagekar	Diva	9640250120

Cont.

Second normal form

It is in 1NF and all non-key attributes are fully functional dependent on the primary key

StudentCourse

StudentNo	CourseNo	Grade
101	C201	A
102	C201	A+
103	C201	B
104	C201	A
105	C201	B
106	C201	A+
107	C202	B
108	C202	B+
109	C202	C
110	C202	A+
111	C203	A
112	C203	A
113	C203	B
114	C203	B+
115	C203	A
101	C202	A
103	C203	A+
105	C202	A
107	C201	A
110	C203	B

Coursedetails

CourseNo	CourseName	InstructorNo	InstructorNam	InstructorLocation	Fees
C201	Web Development	IN101	Tejraj Singh Hada	Thane	68000
C202	Python Development	IN102	Vipul Kumar	Airoli	72000
C203	Android Development	IN103	Roshni Sharma	Kalyan	87000

How to update 2NF anomalies:

Insertion Anomaly:

When adding a new instructor, we need a course.

Update Anomaly:

Updating course information could lead to inconsistencies for instructor information.

Deletion Anomaly:

Deleting a course may also delete instructor information.

❖ Third Normal Form (3NF):

To be in *third normal form*, the relation must be in second normal form. Also all transitive dependencies must be removed; a non-key attribute may not be functionally dependent on another non-key attribute.

Process for 3NF:

Eliminate all dependent attributes in transitive relationship(s) from each of the tables that have a transitive relationship.

Create new table(s) with removed dependency.

Check new table(s) as well as table(s) modified to make sure that each table has a determinant and that no table contains inappropriate dependencies.

Admin(AdminNo,Admin_Name)

Admin_course(Admin_No,CourseNo)

Admin_Instr(Admin_No,InstructorNo)

Admin_stud(Admin_No,StudentNo)

Student (StudentNo, StudentName,Address,Contact)

StudenEncourse(StudentNo, CourseNo, Grade)

Coursedetails(CourseNo,CourseName, fees)

Instructor (InstructorNo, InstructorName, InstructorLocation)

Third normal form

Admin

Admin_No	Admin_Name
1	Vinayak Ingale
2	Anjali Deshmukh
3	Rutuja Pednekar

Admin_course

Admin_No	CourseNo
1	C201
1	C202
1	C203
2	C201
2	C202
2	C203
3	C201
3	C202
3	C203

Admin_stud

Admin_No	StudentNo
1	101
1	102
1	103
1	104
1	105
1	106
2	107
2	108
2	109
2	110
3	111
3	112
3	113
3	114
3	115

Admin_Instr

Admin_No	InstructorNo
1	IN101
1	IN102
1	IN103
2	IN101
2	IN102
2	IN103
3	IN101
3	IN102
3	IN103

Student

StudentNo	StudentName	Address	Contact
101	Priya Kasalkar	Thane	9896857485
102	Shravani Raut	Kalva	9945125847
103	Sagar Naik	Diva	7785451236
104	Abhishek Kulkarni	Ghansoli	9632145678
105	Diksha Mane	Dombivali	9863524174
106	Sudhir Gurav	Koper	9456321584
107	Ranjita Desai	Thane	9485632158
108	Shweta Rasal	Dombivali	9856743284
109	Minakshi Patil	Diva	9743186392
110	Rasika Deshmukh	Rabale	9482369712
111	Vijaya Dhanavade	Airoli	7758369853
112	Arati sarnaik	Ghatkope	7709220540
113	Seema Uchale	Mulund	7752148250
114	Geeta Patade	Thane	9586241854
115	Gouresh Munagekar	Diva	9640250120

Cont..

Third normal form

if there is no transitive dependency for non-prime attributes as well as it is in second normal form

Student_Enroll_Course

StudentNo	CourseNo	Grade
101	C201	A
102	C201	A+
103	C201	B
104	C201	A
105	C201	B
106	C201	A+
107	C202	B
108	C202	B+
109	C202	C
110	C202	A+
111	C203	A
112	C203	A
113	C203	B
114	C203	B+
115	C203	A
101	C202	A
103	C203	A+
105	C202	A
107	C201	A

Course

CourseNo	CourseName	InstructorNo	Fees
C201	Web Development	IN101	68000
C202	Python Development	IN102	72000
C203	Android Development	IN103	87000

Instructor_Details

InstructorNo	InstructorName	InstructorLocation
IN101	Tejraj Singh Hada	Thane
IN102	Vipul Kumar	Airoli
IN103	Roshni Sharma	Kalyan

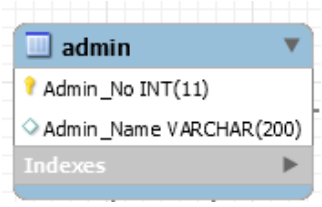
Sheet1

❒ Entity relationship (ER) diagram:

Entity relationship (ER) diagrams can be drawn to show entities and their relationships with one another. These entities can be represented as rectangles as shown below.

The Admin, student, instructor and course tables make up the core of database.

- **Admin:**



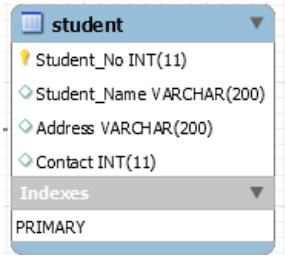
The Admin table, shown above, is used to store basic data about Admin, but it can be expanded according to specific needs. Here the attributes are:

Admin_No-shows Admin number with primary key

Admin_Name-shows Admin Name

Constraint: Admin number will be unique for each Admin table

- **Student:**



The student table, shown above, is used to store basic data about student, but it can be expanded according to specific needs. Here the attributes are:

Student_No-shows student number with primary key

Student_Name-shows Student Name

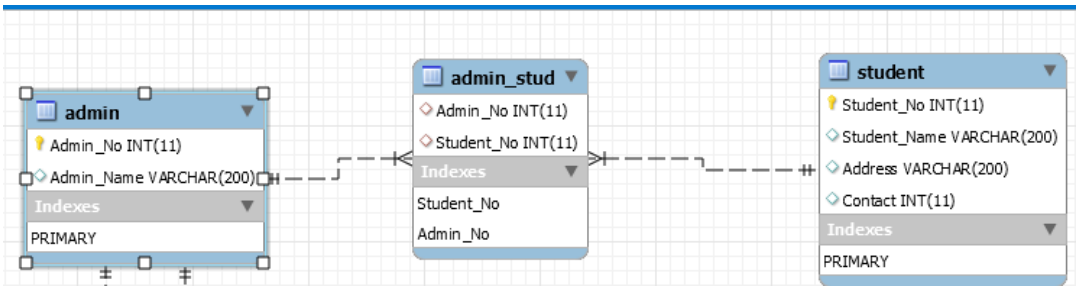
Address-shows address of student

Contact-shows contact number of student

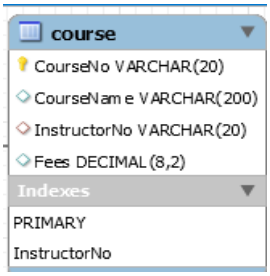
Constraint:Student_No must be unique for all student and the corresponding Admin should exist in Admin table.

Now Admin can manages number of student in institute so here I define relation between them as **Admin manages many student and it is shown as below**

From admin-stud table we get information about admin and his related student



- **Course:**



The Course table, shown above, is used to store basic data about course, but it can be expanded according to specific needs. Here the attributes are:

CourseNo-shows course number with primary key

CourseName-shows course name

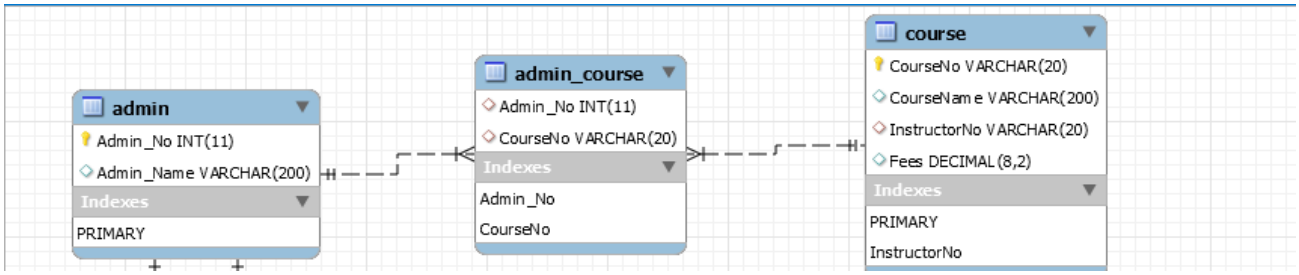
InstructorNo-shows instructor related to this course

Fees-shows fees related with course

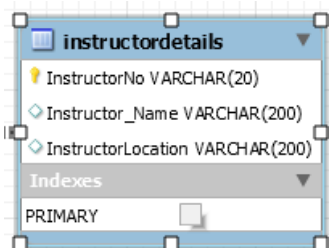
Constraint: CourseNo will be unique for each Course and the corresponding Admin and Instructor should exist in Instructor table

Here in this table InstructorNo is foreign key from instructor table

Also **Admin can manage many courses** like add course, remove course, update fees of particular course so here define relation between Admin and course.



- **Instructor:**



The Instructor table, shown above, is used to store basic data about instructor, but it can be expanded according to specific needs. Here the attributes are:

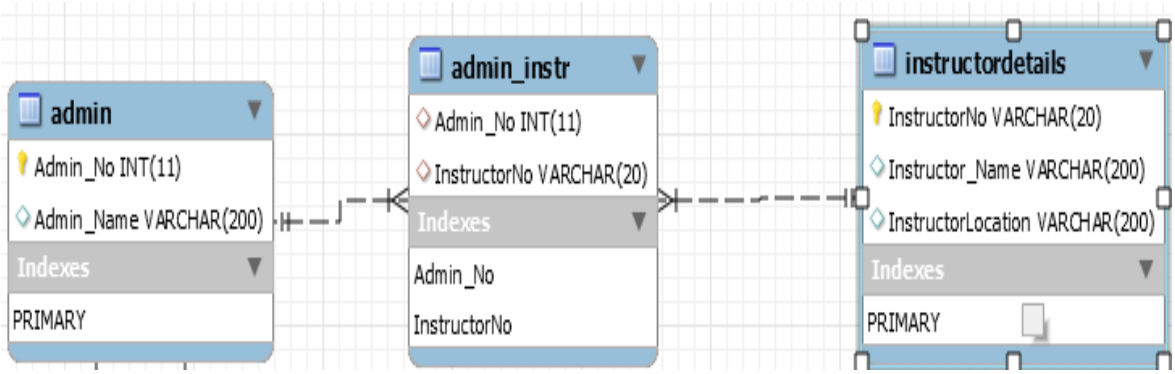
InstructorNo-shows instructor number with primary key.

Instructor_Name –shows instructor name

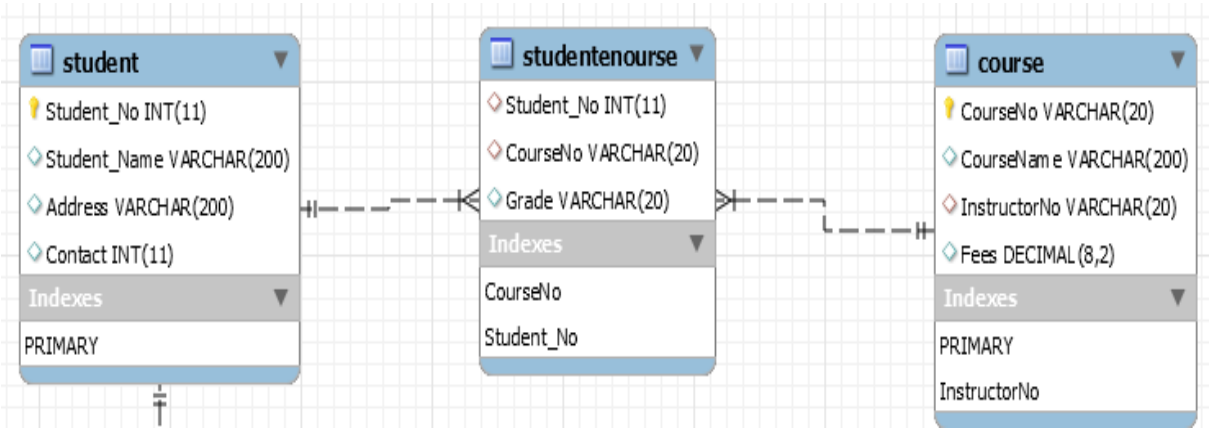
InstructorLocation-shows Instructor location

Constraint:InstructorNo will be unique for each Instructor

Also **Admin can manage many Instructors** like add new Instructor, remove Instructor and update information of Instructor like location so here define relation between Admin and Instructor.



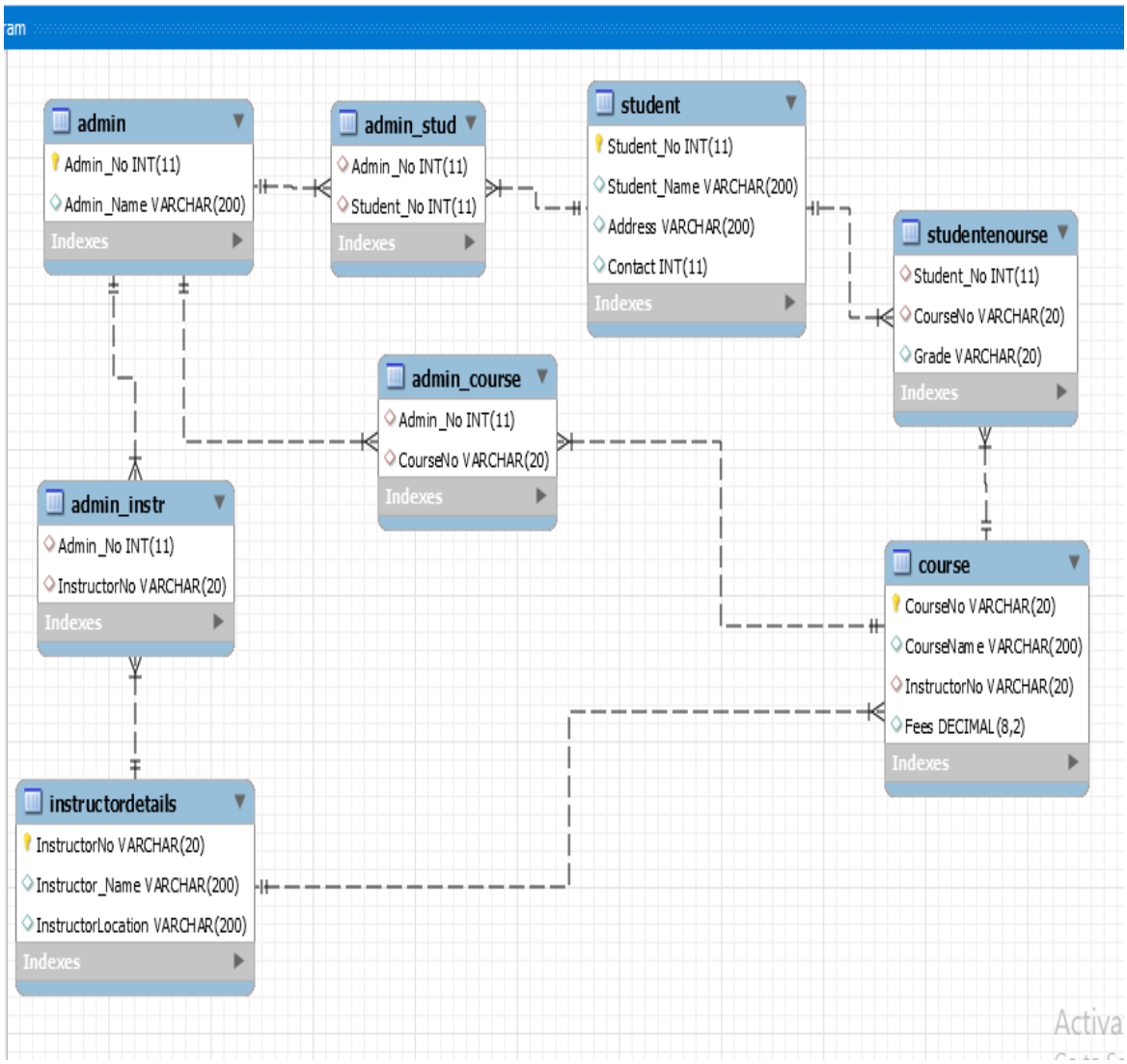
❖ **Relation between student and course:**



In above diagram **one student related to many courses**

From studentenourse table we can find perticular course information with grade by using Student_No

❖ **ER Diagram:**



❒ Creating Roles:

A role is created to ease setup and maintenance of the security model. It is a named group of related privileges that can be granted to the user. When there are many users in a database it becomes difficult to grant or revoke privileges to users.

Therefore, if you define roles:You can grant or revoke privileges to users, thereby automatically granting or revoking privileges.

❖ Create Role and user:

Here in this database first **create role as Admin With user works as Admin**

1) Vinayak Ingale

2) Anjali Deshmukh

3) Rutuja Pednekar

```
MariaDB [(none)]> create database Coaching_Institute_manage;
Query OK, 1 row affected (0.126 sec)

MariaDB [(none)]>  use Coaching_Institute_manage;
Database changed
MariaDB [Coaching_Institute_manage]> create role Admin;
Query OK, 0 rows affected (0.638 sec)
```

Now grant privillages to Admin:

```
MariaDB [Coaching_Institute_manage]> grant create,insert,update,alter,drop,select,create view,delete on Coaching_Institute
_manage.* to Admin;
Query OK, 0 rows affected (0.069 sec)
```

Here Admin can Create table,insert records into table,update table record,alter table structure,select data from table,drop table

```
MariaDB [Coaching_Institute_manage]> create user 'Vinayak_Ingale'@'localhost' identified by 'Vinayak_ingale';
Query OK, 0 rows affected (0.296 sec)

MariaDB [Coaching_Institute_manage]> grant Admin to 'Vinayak_Ingale'@'localhost';
Query OK, 0 rows affected (0.055 sec)

MariaDB [Coaching_Institute_manage]> show grants for 'Vinayak_Ingale'@'localhost';
+-----------------------------------------------------------------------------------------------------------------------+
| Grants for Vinayak_Ingale@localhost                                                                                   |
+-----------------------------------------------------------------------------------------------------------------------+
| GRANT `Admin` TO `Vinayak_Ingale`@`localhost`                                                                         |
| GRANT USAGE ON *.* TO `Vinayak_Ingale`@`localhost` IDENTIFIED BY PASSWORD '*F0D8EDC6754D36704F28E4B0922DF222D4417584' |
+-----------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.001 sec)

MariaDB [Coaching_Institute_manage]> exit
Bye
```

Now User as Admin can login with given user name and password.

```
D:\xampp\mysql\bin>mysql -u Vinayak_Ingale -p
Enter password: **************
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
Server version: 10.4.17-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
2 rows in set (0.072 sec)

MariaDB [(none)]> set role Admin;
Query OK, 0 rows affected (0.000 sec)
```

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
2 rows in set (0.001 sec)

MariaDB [(none)]> set role Admin;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> show databases;
+---------------------------+
| Database                  |
+---------------------------+
| coaching_institute_manage |
| information_schema        |
| test                      |
+---------------------------+
3 rows in set (0.002 sec)
```

Create another user with Admin role

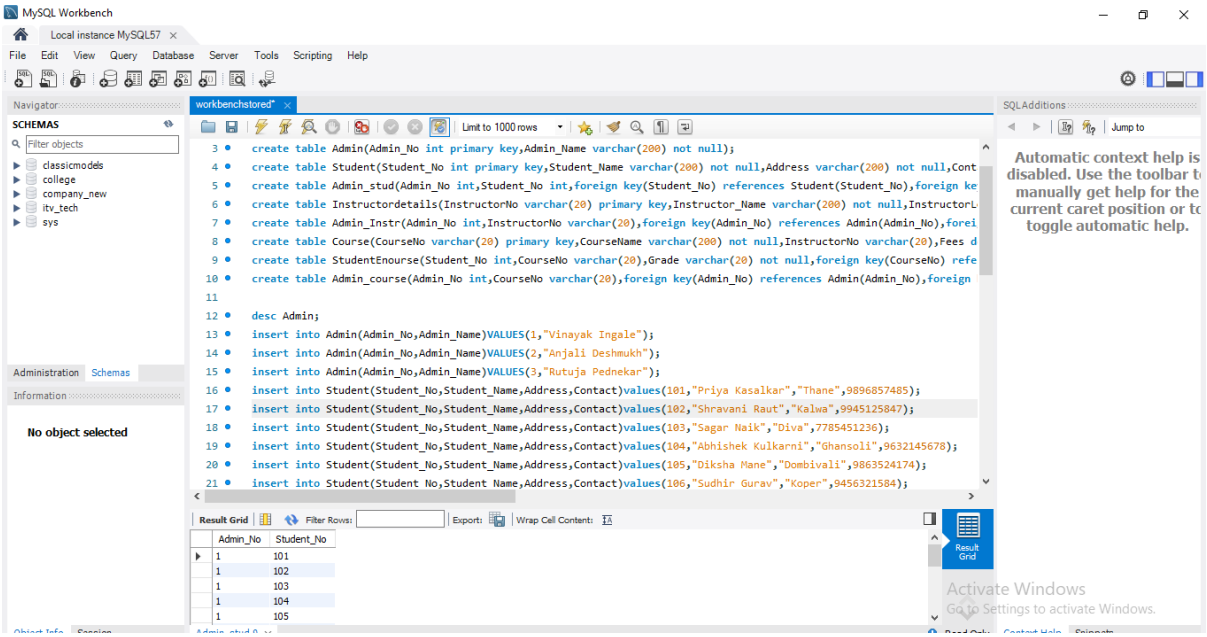
```
MariaDB [(none)]>  use Coaching_Institute_manage;
Database changed
MariaDB [Coaching_Institute_manage]> create user 'Anjali_Deshmukh'@'localhost' identified by 'Anjali_Deshmukh';
Query OK, 0 rows affected (0.070 sec)

MariaDB [Coaching_Institute_manage]> grant Admin to 'Anjali_Deshmukh'@'localhost';
Query OK, 0 rows affected (0.052 sec)

MariaDB [Coaching_Institute_manage]> show grants for 'Anjali_Deshmukh'@'localhost';
+-----------------------------------------------------------------------------------------------------------------------------------+

| Grants for Anjali_Deshmukh@localhost                                                                                              |

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

| GRANT `Admin` TO `Anjali_Deshmukh`@`localhost`                                                                                    |

| GRANT USAGE ON *.* TO `Anjali_Deshmukh`@`localhost` IDENTIFIED BY PASSWORD '*D6235424F5D51A104128D549A93258250CB458C8' |

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

2 rows in set (0.000 sec)

MariaDB [Coaching_Institute_manage]> exit
Bye
```

Now user can create table because he has privillage to create table.

```
MariaDB [(none)]> show databases;
+---------------------------+
| Database                  |
+---------------------------+
| coaching_institute_manage |
| information_schema        |
| test                      |
+---------------------------+
3 rows in set (0.001 sec)

MariaDB [(none)]>  use Coaching_Institute_manage;
Database changed
MariaDB [Coaching_Institute_manage]> create table Admin(Admin_No int primary key,Admin_Name varchar(200) not null);
Query OK, 0 rows affected (0.530 sec)
```

```
MariaDB [Coaching_Institute_manage]> desc Student;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| Student_No   | int(11)      | NO   | PRI | NULL    |       |
| Student_Name | varchar(200) | NO   |     | NULL    |       |
| Address      | varchar(200) | NO   |     | NULL    |       |
| Contact      | int(11)      | NO   |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
4 rows in set (0.085 sec)

MariaDB [Coaching_Institute_manage]> drop table Student;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
MariaDB [Coaching_Institute_manage]> alter table Student change Contact varchar(200);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version
 for the right syntax to use near 'varchar(200)' at line 1
MariaDB [Coaching_Institute_manage]> alter table Student modify column Contact varchar(200);
Query OK, 15 rows affected (1.621 sec)
Records: 15  Duplicates: 0  Warnings: 0

MariaDB [Coaching_Institute_manage]> create table Student(Stdent_No int,Student_Name varchar(200),Address varchar(200),Con
tact varchar(200));
Query OK, 0 rows affected (0.858 sec)
MariaDB [Coaching_Institute_manage]>
```

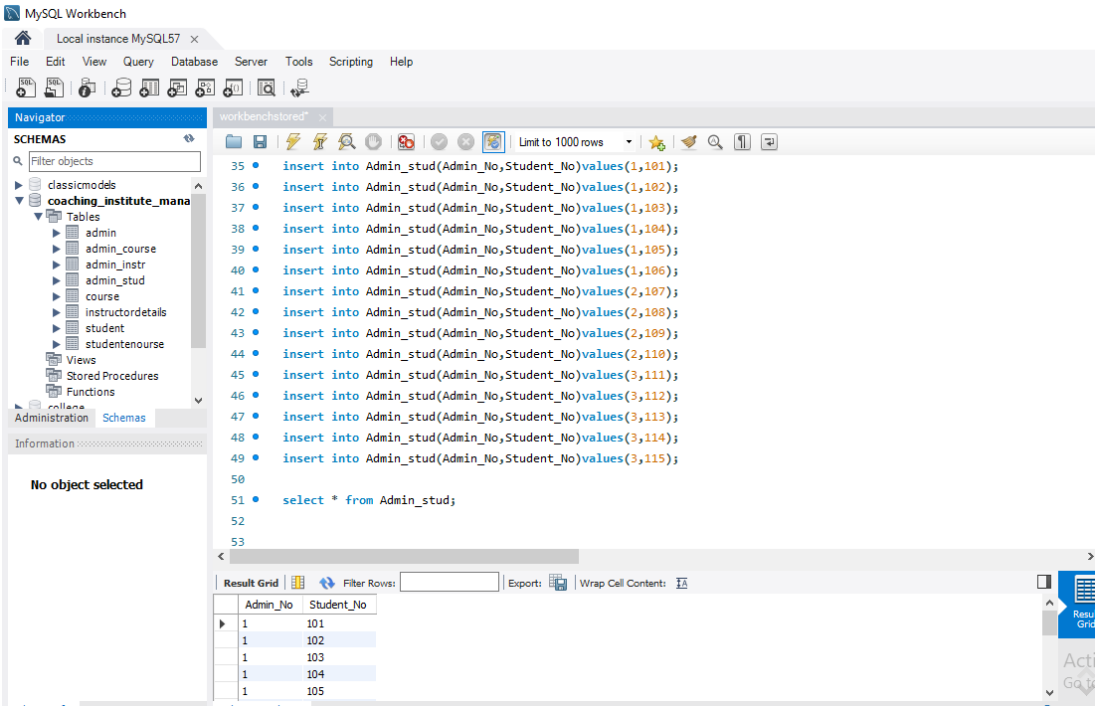
Or we can create table in Mysql workbench



Insert data into tables:

```
MariaDB [Coaching_Institute_manage]> insert into Student(Student_No,Student_Name,Address,Contact)values(101,"Priya Kasalka
r","Thane",9896857485);
Query OK, 1 row affected (0.057 sec)

MariaDB [Coaching_Institute_manage]> insert into Student(Student_No,Student_Name,Address,Contact)values(102,"Shravani Raut
","Kalwa",9945125847);
Query OK, 1 row affected (0.098 sec)

MariaDB [Coaching_Institute_manage]> insert into Student(Student_No,Student_Name,Address,Contact)values(103,"Sagar Naik","
Diva",7785451236);
Query OK, 1 row affected (0.063 sec)

MariaDB [Coaching_Institute_manage]> insert into Student(Student_No,Student_Name,Address,Contact)values(104,"Abhishek Kulk
arni","Ghansoli",9632145678);
Query OK, 1 row affected (0.134 sec)

MariaDB [Coaching_Institute_manage]> insert into Student(Student_No,Student_Name,Address,Contact)values(105,"Diksha Mane",
"Dombivali",9863524174);
Query OK, 1 row affected (0.054 sec)

MariaDB [Coaching_Institute_manage]> insert into Student(Student_No,Student_Name,Address,Contact)values(106,"Sudhir Gurav"
,"Koper",9456321584);
Query OK, 1 row affected (0.093 sec)

MariaDB [Coaching_Institute_manage]> insert into Student(Student_No,Student_Name,Address,Contact)values(107,"Ranjita Desai
","Thane",9485632158);
Query OK, 1 row affected (0.118 sec)

MariaDB [Coaching_Institute_manage]> insert into Student(Student_No,Student_Name,Address,Contact)values(108,"Shweta Rasal"
```

Insert data into Admin_stud



In Below scenario Every Admin can Manage every Instructor.

```
insert into Admin_Instr(Admin_No,InstructorNo)values(1,"In101");
insert into Admin_Instr(Admin_No,InstructorNo)values(1,"In102");
insert into Admin_Instr(Admin_No,InstructorNo)values(1,"In103");
insert into Admin_Instr(Admin_No,InstructorNo)values(2,"In101");
insert into Admin_Instr(Admin_No,InstructorNo)values(2,"In102");
insert into Admin_Instr(Admin_No,InstructorNo)values(2,"In103");
insert into Admin_Instr(Admin_No,InstructorNo)values(3,"In101");
insert into Admin_Instr(Admin_No,InstructorNo)values(3,"In102");
insert into Admin_Instr(Admin_No,InstructorNo)values(3,"In103");
select * from Admin_Instr;
```

Admin_No	InstructorNo
1	In101
1	In102
1	In103
2	In101
2	In102

Show all tables:

Student:

```
select * from Student;
select * from Admin;
select * from Course;
select * from Instructordetails;
```

Student_No	Student_Name	Address	Contact
101	Priya Kasalkar	Thane	9896857485
102	Shravani Raut	Kalwa	9945125847
103	Sagar Naik	Diva	7785451236
104	Abhishek Kulkarni	Ghansoli	9632145678
105	Diksha Mane	Dombivali	9863524174
106	Sudhir Gurav	Koper	9456321584
107	Ranjita Desai	Thane	9485632158
108	Shweta Rasal	Dombivali	9856743284
109	Minakshi Patil	Diva	9743186392
110	Rasika Deshmukh	Rabale	9482369712
111	Vijaya Dhanavade	Airoli	7758369853
112	Arati sarnaik	Ghatkoper	7709220540
113	Seema Uchale	Mulund	7752148250
114	Geeta Patade	Thane	9586241854
115	Gouresh Munag...	Diva	9640250120

Admin:

```
select * from Admin;
```

Admin_No	Admin_Name
1	Vinayak Ingale
2	Anjali Deshmukh
3	Rutuja Pednekar

Course:

```
107 •  select * from Course;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

CourseNo	CourseName	InstructorNo	Fees
C201	Web Development	IN101	68000.00
C202	Python Development	IN102	72000.00
C203	Android Development	IN103	87000.00
NULL	NULL	NULL	NULL

Instructordetails:

```
108 •  select * from Instructordetails;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

InstructorNo	Instructor_Name	InstructorLocation
IN101	Tejraj Singh Hada	Thane
IN102	Vipul Kumar	Airoli
IN103	Roshni Sharma	Kalyan
NULL	NULL	NULL

Admin_stud:

```
109 •  select * from Admin_stud;
```

Result Grid | Filter Rows: | Exp

Admin_No	Student_No
1	101
1	102
1	103
1	104
1	105
1	106
2	107
2	108
2	109
2	110
3	111
3	112
3	113
3	114
3	115

Admin_Instr:

```
111 •  select * from Admin_Instr;
```

Result Grid | Filter Rows: | Export:

Admin_No	InstructorNo
1	In101
1	In102
1	In103
2	In101
2	In102
2	In103
3	In101
3	In102
3	In103

Admin_course:

```
112 •  select * from Admin_course;
```

Result Grid | Filter Rows: | Export:

Admin_No	CourseNo
1	C201
1	C202
1	C203
2	C201
2	C202
2	C203
3	C201
3	C202

StudentEnourse:

```
110 •  select * from StudentEnourse;
```

Result Grid | Filter Rows: | Export:

Student_No	CourseNo	Grade
101	C201	A
102	C201	A+
103	C201	B
104	C201	A
105	C201	B
106	C201	A+
107	C202	B
108	C202	B+
109	C202	C
110	C202	A+
111	C203	A
112	C203	A
113	C203	B
114	C203	B+
115	C203	A
101	C202	A
103	C203	A+
105	C202	A
107	C201	A
110	C203	B

Create role Student:

Student view Course,Student and Instructordetails by using select privillages.

Student can update his/her information so add update privillage on Student table

```
MariaDB [Coaching_Institute_manage]> create role student;
Query OK, 0 rows affected (0.173 sec)

MariaDB [Coaching_Institute_manage]> grant select on Coaching_Institute_manage.Course to student;
Query OK, 0 rows affected (0.025 sec)

MariaDB [Coaching_Institute_manage]> grant select on Coaching_Institute_manage.Student to student;
Query OK, 0 rows affected (0.067 sec)

MariaDB [Coaching_Institute_manage]> grant update on Coaching_Institute_manage.Student to student;
Query OK, 0 rows affected (0.021 sec)

MariaDB [Coaching_Institute_manage]> grant select on Coaching_Institute_manage.Instructordetails to student;
Query OK, 0 rows affected (0.055 sec)

MariaDB [Coaching_Institute_manage]> create user 'Priya_Kasalkar'@'localhost' identified by 'Priya_k';
Query OK, 0 rows affected (0.064 sec)

MariaDB [Coaching_Institute_manage]> grant student to 'Priya_Kasalkar'@'localhost';
Query OK, 0 rows affected (0.050 sec)

MariaDB [Coaching_Institute_manage]> exit
Bye
```

```
D:\xampp\mysql\bin>mysql -u Priya_Kasalkar -p
Enter password: *******
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 60
Server version: 10.4.17-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> set role student;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> show databases;
+--------------------------+
| Database                 |
+--------------------------+
| coaching_institute_manage |
| information_schema       |
| test                     |
+--------------------------+
3 rows in set (0.056 sec)

MariaDB [(none)]> use Coaching_Institute_manage;
Database changed
MariaDB [Coaching_Institute_manage]> create table new_course(couseid int,course_name varchar(20));
ERROR 1142 (42000): CREATE command denied to user 'Priya_Kasalkar'@'localhost' for table 'new_course'
```



```
MariaDB [Coaching_Institute_manage]> select * from Student;
+------------+------------------+-----------+------------+
| Student_No | Student_Name     | Address   | Contact    |
+------------+------------------+-----------+------------+
|        101 | Priya Kasalkar   | Thane     | 9896857485 |
|        102 | Shravani Raut    | Kalwa     | 9945125847 |
|        103 | Sagar Naik       | Diva      | 7785451236 |
|        104 | Abhishek Kulkarni | Ghansoli  | 9632145678 |
|        105 | Diksha Mane      | Dombivali | 9863524174 |
|        106 | Sudhir Gurav     | Koper     | 9456321584 |
|        107 | Ranjita Desai    | Thane     | 9485632158 |
|        108 | Shweta Rasal     | Dombivali | 9856743284 |
|        109 | Minakshi Patil   | Diva      | 9743186392 |
|        110 | Rasika Deshmukh  | Rabale    | 9482369712 |
|        111 | Vijaya Dhanavade | Airoli    | 7758369853 |
|        112 | Arati sarnaik    | Ghatkoper | 7709220540 |
|        113 | Seema Uchale     | Mulund    | 7752148250 |
|        114 | Geeta Patade     | Thane     | 9586241854 |
|        115 | Gouresh Munagekar | Diva      | 9640250120 |
+------------+------------------+-----------+------------+
15 rows in set (0.072 sec)

MariaDB [Coaching_Institute_manage]> select * from Admin_course;
ERROR 1142 (42000): SELECT command denied to user 'Priya_Kasalkar'@'localhost' for table 'admin_course'
MariaDB [Coaching_Institute_manage]>
```

In above scenario with student role we can select all details of Student table because Priya_Kasalkar is a user with student role have privillages of view all details of course,Instructor and update on Student table so he/she can update own information

Create role Instructor:

Instructor view Course and Instructordetails,Admin,Student by using select privillage.

Instructor can update his/her information so add update privillage on Instructor table

and StudentEnourse table to add grade.

```
MariaDB [Coaching_Institute_manage]> create role Instructor;
Query OK, 0 rows affected (0.236 sec)
```

```
MariaDB [Coaching_Institute_manage]> grant select on Coaching_Institute_manage.Course to Instructor;
Query OK, 0 rows affected (0.089 sec)

MariaDB [Coaching_Institute_manage]> grant select on Coaching_Institute_manage.Student to Instructor;
Query OK, 0 rows affected (0.064 sec)

MariaDB [Coaching_Institute_manage]> grant select on Coaching_Institute_manage.Instructordetails to Instructor;
Query OK, 0 rows affected (0.066 sec)

MariaDB [Coaching_Institute_manage]> grant update on Coaching_Institute_manage.Instructordetails to Instructor;
Query OK, 0 rows affected (0.021 sec)

MariaDB [Coaching_Institute_manage]> grant update on Coaching_Institute_manage.StudentEnourse to Instructor;
Query OK, 0 rows affected (0.049 sec)

MariaDB [Coaching_Institute_manage]> grant select on Coaching_Institute_manage.Admin to Instructor;
Query OK, 0 rows affected (0.177 sec)
```



```
MariaDB [Coaching_Institute_manage]> create user 'Tejraj'@'localhost' identified by 'Tejraj_h';
Query OK, 0 rows affected (0.079 sec)

MariaDB [Coaching_Institute_manage]> grant Instructor to 'Tejraj'@'localhost';
Query OK, 0 rows affected (0.051 sec)
```

```
Select Command Prompt - mysql -u Tejraj -p
MariaDB [Coaching_Institute_manage]> show grants for 'Tejraj'@'localhost';
+-----------------------------------------------------------------------------------------------------------------+
| Grants for Tejraj@localhost                                                                                     |
+-----------------------------------------------------------------------------------------------------------------+
| GRANT `Instructor` TO `Tejraj`@`localhost`                                                                      |
| GRANT USAGE ON *.* TO `Tejraj`@`localhost` IDENTIFIED BY PASSWORD '*D42E6F5BDBD41180C2E289BAA01A5F4F9B587987' |
+-----------------------------------------------------------------------------------------------------------------+
2 rows in set (0.000 sec)

MariaDB [Coaching_Institute_manage]> exit
Bye

D:\xampp\mysql\bin>mysql -u Tejraj -p
Enter password: ********
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 66
Server version: 10.4.17-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> set role Instructor;
Query OK, 0 rows affected (0.000 sec)
```

Select Command Prompt - mysql -u Tejraj -p

```
MariaDB [(none)]> show databases;
+--------------------------+
| Database                 |
+--------------------------+
| coaching_institute_manage |
| information_schema       |
| test                     |
+--------------------------+
3 rows in set (0.116 sec)

MariaDB [(none)]> use Coaching_Institute_manage;
Database changed
```

```
MariaDB [Coaching_Institute_manage]> select * from Course;
+----------+--------------------+--------------+----------+
| CourseNo | CourseName         | InstructorNo | Fees     |
+----------+--------------------+--------------+----------+
| C201     | Web Development    | IN101        | 68000.00 |
| C202     | Python Development | IN102        | 72000.00 |
| C203     | Android Development | IN103        | 87000.00 |
+----------+--------------------+--------------+----------+
3 rows in set (0.078 sec)

MariaDB [Coaching_Institute_manage]> select * from Instructordetails;
+--------------+------------------+--------------------+
| InstructorNo | Instructor_Name  | InstructorLocation |
+--------------+------------------+--------------------+
| IN101        | Tejraj Singh Hada | Thane              |
| IN102        | Vipul Kumar      | Airoli             |
| IN103        | Roshni Sharma    | Kalyan             |
+--------------+------------------+--------------------+
3 rows in set (0.001 sec)
```

```
MariaDB [Coaching_Institute_manage]> update Instructordetails set InstructorLocation="Ghansoli" where InstructorNo='In101'
;
Query OK, 1 row affected (0.360 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Coaching_Institute_manage]> select * from Instructordetails;
+--------------+------------------+--------------------+
| InstructorNo | Instructor_Name  | InstructorLocation |
+--------------+------------------+--------------------+
| IN101        | Tejraj Singh Hada | Ghansoli           |
| IN102        | Vipul Kumar      | Airoli             |
| IN103        | Roshni Sharma    | Kalyan             |
+--------------+------------------+--------------------+
3 rows in set (0.001 sec)
```

```
MariaDB [Coaching_Institute_manage]> update Course set CourseName="Java" where Courseno='C101';
ERROR 1142 (42000): UPDATE command denied to user 'Tejraj'@'localhost' for table 'course'
```

```
MariaDB [Coaching_Institute_manage]> select * from Admin;
+----------+----------------+
| Admin_No | Admin_Name     |
+----------+----------------+
|        1 | Vinayak Ingale |
|        2 | Anjali Deshmukh |
|        3 | Rutuja pednekar |
+----------+----------------+
3 rows in set (0.038 sec)
```

Queries:

1)Find the name of course which has fess equal to 72000

```
select CourseName from Course where Fees=72000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CourseName
Python Development

2) Write a query to change Grade column size by200

```
desc StudentEnourse;
alter table StudentEnouse modify column Grade varchar(200);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Student_No	int(11)	YES	MUL	NULL	
CourseNo	varchar(20)	YES	MUL	NULL	
Grade	varchar(20)	NO		NULL	

```
alter table StudentEnourse modify column Grade varchar(200);
desc StudentEnourse;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Student_No	int(11)	YES	MUL	NULL	
CourseNo	varchar(20)	YES	MUL	NULL	
Grade	varchar(200)	YES		NULL	

3)Find the Student_No of all students who were taught by an instructor named Tejraj; make sure there are no duplicates in the result.

```
select distinct Student_No,Student_Name from Student join StudentEnourse using(Student_No) join Course using(CourseNo)
join Instructordetails using(InstructorNo) where Instructor_name = "Tejraj Singh Hada";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Student_No	Student_Name
101	Priya Kasalkar
102	Shravani Raut
103	Sagar Naik
104	Abhishek Kulkarni
105	Diksha Mane
106	Sudhir Gurav
107	Ranjita Desai

4) Create a view name student containing Student number,name and location and sort student number in descending order.

```
create view Studentview as select Student_No,Student_Name,Address,Contact from Student;
select * from Studentview order by Student_No desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Student_No	Student_Name	Address	Contact
115	Gouresh Munagekar	Diva	9640250120
114	Geeta Patade	Thane	9586241854
113	Seema Uchale	Mulund	7752148250
112	Arati sarnaik	Ghatkoper	7709220540
111	Vijaya Dhanavade	Airoli	7758369853
110	Rasika Deshmukh	Rabale	9482369712
109	Minakshi Patil	Diva	9743186392
108	Shweta Rasal	Dombivali	9856743284
107	Ranjita Desai	Thane	9485632158
106	Sudhir Gurav	Koper	9456321584
105	Diksha Mane	Dombivali	9863524174
104	Abhishek Kulkarni	Ghansoli	9632145678
103	Sagar Naik	Diva	7785451236
102	Shravani Raut	Kalwa	9945125847
101	Priya Kasalkar	Thane	9896857485

5)show Student grade with the help of student number and course number using stored procedure

```
152
153 • select Grade from StudentEnourse where Student_No=107 and CourseNo="C202";
154   delimiter $$
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Grade
B

Limit to 1000 rows

```
153
154
155   delimiter $$
156 • create procedure CourseGradenew45(in studno int,in course varchar(200))
157   begin
158       select Grade from StudentEnourse where Student_No=studno and CourseNo=course;
159   end $$
160   delimiter ;
161 • call CourseGradenew45(107,"C202")
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Grade
B