Online Learning Platforms

Problem Statement

2020 saw a sudden rise in the growth of people on the internet, the reason being Covid-19. More people online essentially means more people willing to help communities. Ideas were discussed , stories were narrated, jokes and memes were shared. Of the many meme topics, one that stayed for a while was on Indian educators. How people around the world shared a common experience of being 'saved' by Indian educators at the last moment before exams. How they helped many around the world to gain knowledge at a time when it mattered the most.

Gone are the days when people would stick to the things they are taught in classrooms. Many would rather study something else, many don't understand what they are being taught, some want to get their hands dirty on more practical applications of things, while some would also want to learn something not available in their area.

The problems, though varied, can be tackled by online learning platforms. A place where people can learn things they want to learn, from instructors they understand better from and at their own pace from the comfort of their homes. A place where learning is prioritized beyond teachers, institutions and books. A new way of learning for a new and rapidly changing world.
Learning can be done at remote locations from some of the best subject experts at a reasonable cost. Instructors from around the world help students learn what they know best. Platforms such as these also incentivize teachers as even they can teach according to their way.

Furthermore communities forming on these platforms have students around the world who can help each other with each other's doubts.
Some good examples : Udemy, Coursera

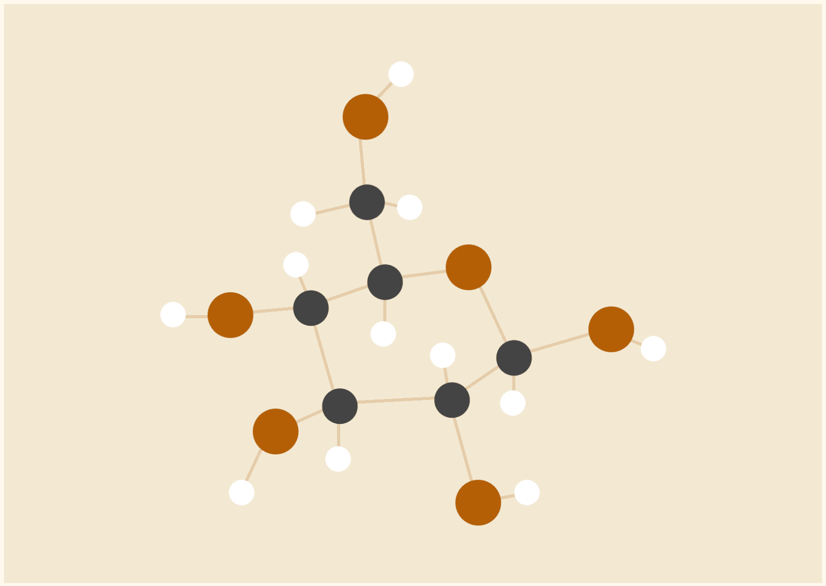
An e-learning platform can benefit from a database management system (DBMS) by improving the organization, accessibility, and security of data. A DBMS can store and manage large amounts of data related to courses, student information, progress tracking, etc. It can provide effective search and retrieval of data, which can help improve the user experience. With a DBMS, an e-learning platform can manage data efficiently, make data-driven decisions, and provide personalized learning experiences to users.

Improves data organization and retrieval for efficient decision-making.
Supports personalized learning experiences through data analysis and student tracking. Provides scalability for handling large amounts of data.
Enhances accessibility by allowing authorized users to access data from anywhere. Ensures data security through authentication and encryption.

This database would contain information of students , like the courses they have purchased, topics they are interested in, wishlist and other basic details. Furthermore it would also contain information regarding instructors like the courses offered by them, subjects they teach ,ratings, their education background etc. Courses on the platform will also naturally be divided according to their respective fields and would contain other relevant data like language offered in, length of course, number of students enrolled , average rating , cost etc.

DBMS ER Model

Online Learning Platform



Members

Name : Aman Seervi

Roll No : 21CSB0B01

Section : B

Name: Naman Mulani

Roll No : 21CSB0B38

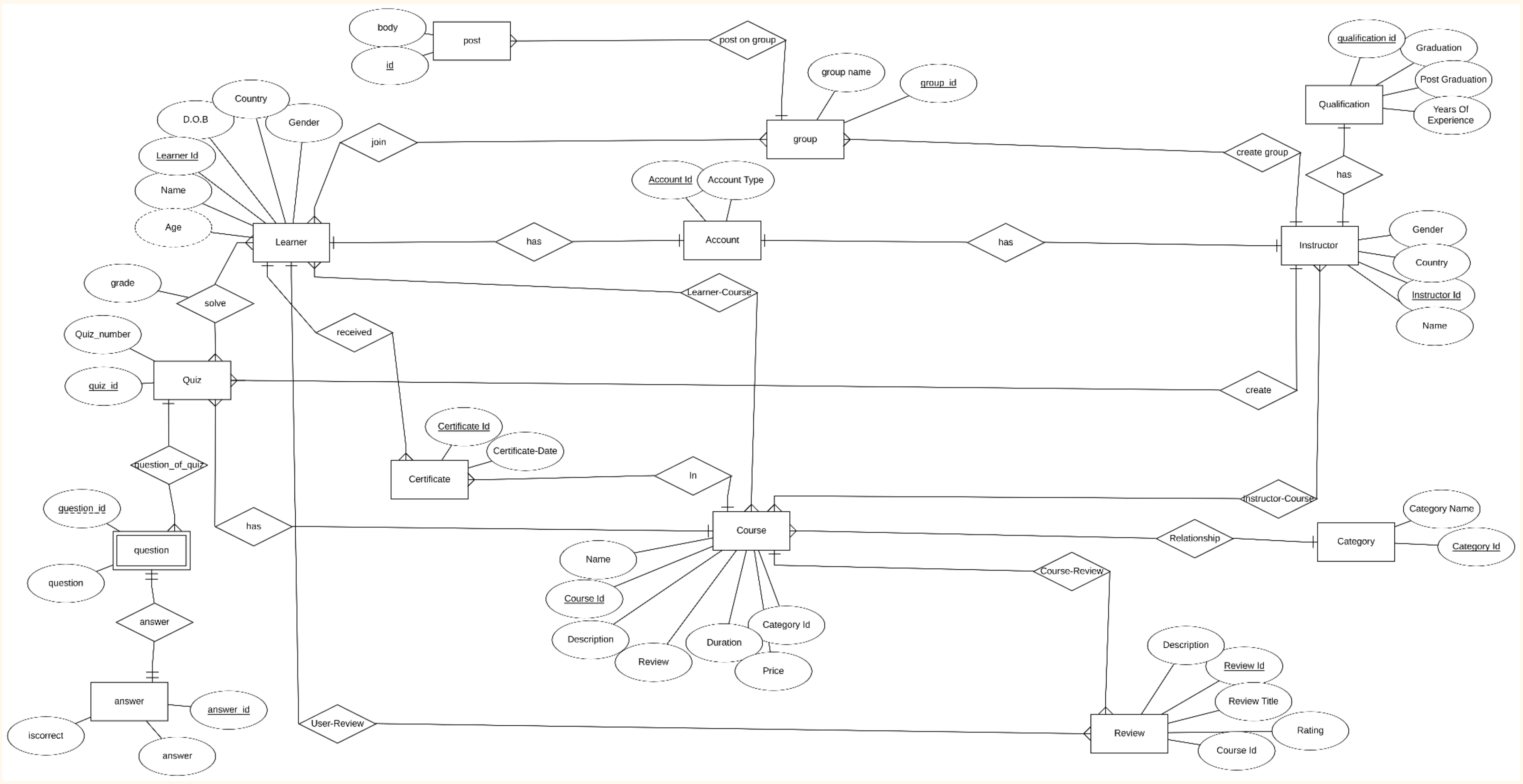
Section : B

Supervised By :

Dr.R.B.V.Suramaanyam

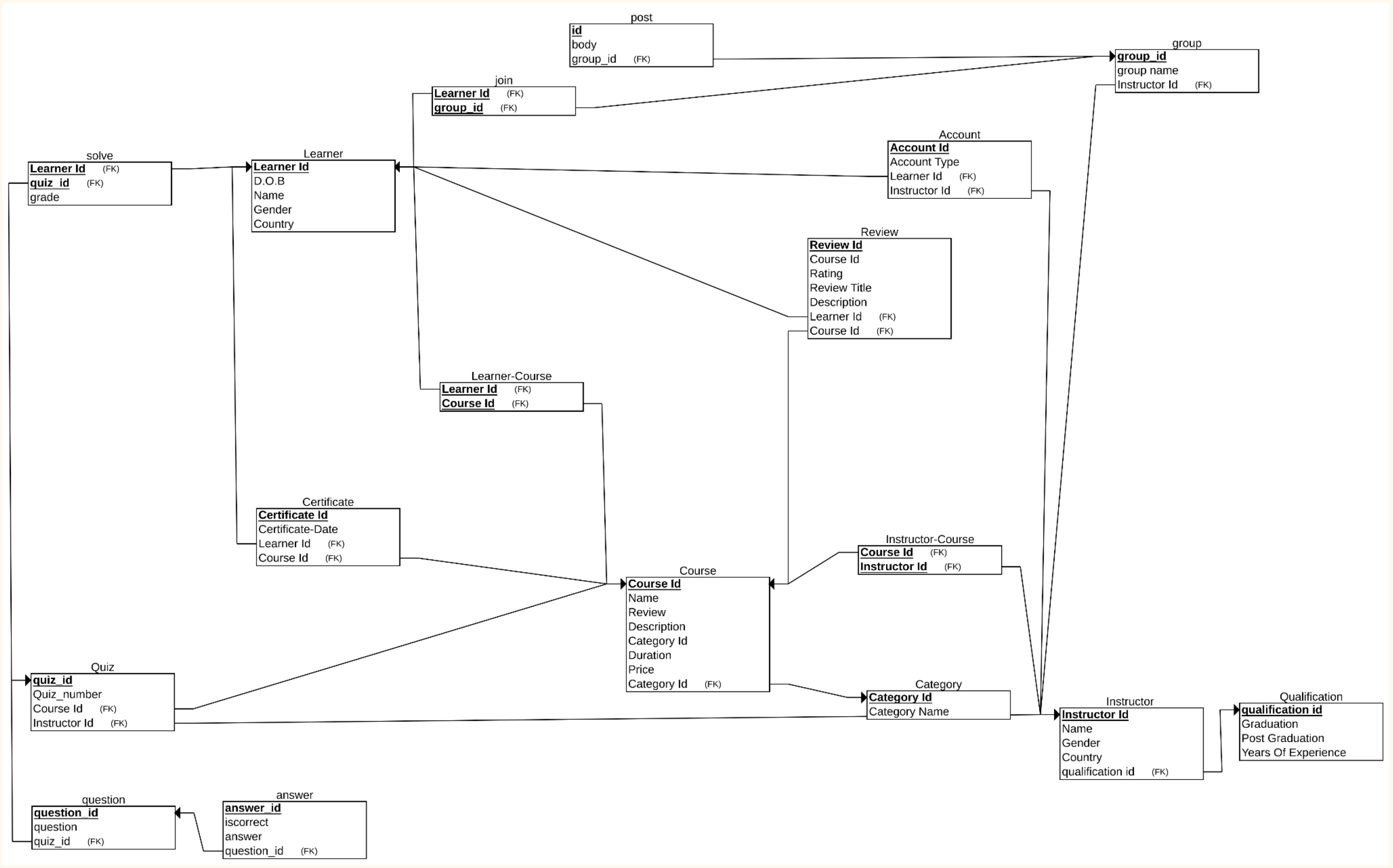
Dr.T.Ramakrishnandu

ER Model



post
body
id
post on group
group
group name
group_id
join
create group
Country
D.O.B
Gender
Learner Id
Name
Age
Learner
has
Account
Account Id
Account Type
has
Instructor
Gender
Country
Instructor Id
Name
qualification id
Graduation
Post Graduation
Years Of Experience
Qualification
has
Learner-Course
grade
solve
received
Quiz_number
quiz_id
Quiz
create
Certificate Id
Certificate-Date
Certificate
In
question_of_quiz
Instructor-Course
question_id
question
question
has
Course
Name
Course Id
Description
Review
Duration
Category Id
Price
Relationship
Category
Category Name
Category Id
Course-Review
answer
answer
answer_id
iscorrect
answer
User-Review
Review
Description
Review Id
Review Title
Rating
Course Id

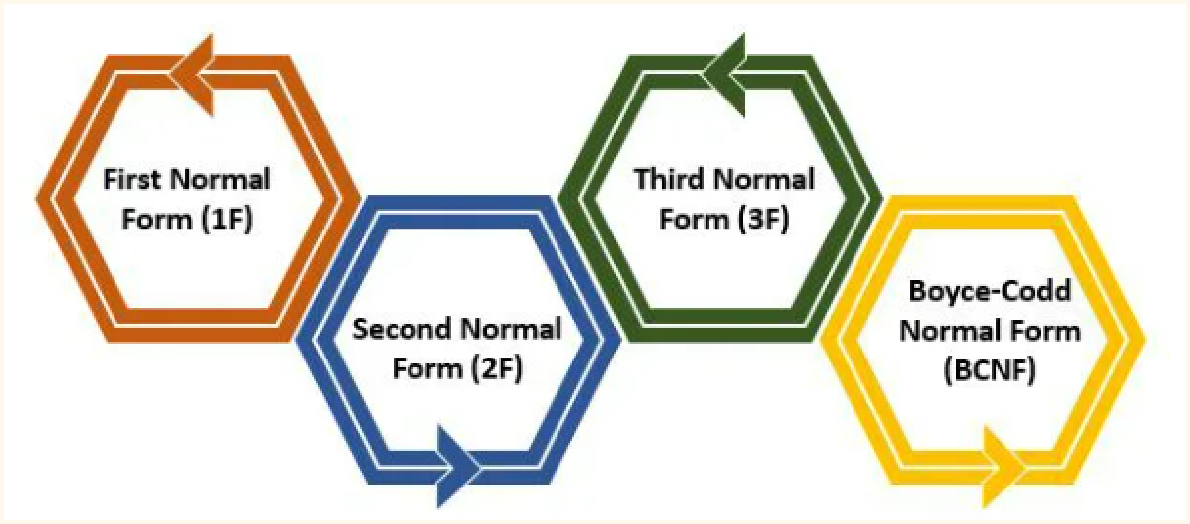
Relational Schema



Assumptions

1. An account can either belong to a learner or an instructor.
2. A learner can buy many courses. Also a particular course can be bought by many distinct learners.
3. A learner can receive many certificates, but a particular certificate can only belong to one learner who has specialized in that corresponding course.
4. A single course might yield a lot of learners a certificate but once again a particular certificate can only belong to a single course.
5. An instructor can create many courses but a particular course can only belong to one instructor.
6. Many courses can belong to a certain category but a single course can only belong to a particular category.
7. A course can be reviewed by many learners but a particular review can only be shared corresponding to a particular course.
8. A learner can share many reviews across courses but a particular review only belongs to a single unique learner.
9. We assume that each course has a group associated with it and learners may opt to join it or not.
10. An instructor can create multiple courses and hence can naturally be part of multiple groups.
11. A group can have many learners.
12. A group can have multiple posts, while a particular post can only belong to a single group
13. One learner can take multiple quizzes and a quiz is taken by many different learners.
14. One instructor can create many quizzes, while a particular quiz can only belong to a particular instructor.
15. One quiz can have many questions, while a particular question can only belong to one quiz.
16. A question can and must have only one answer, and vice versa.

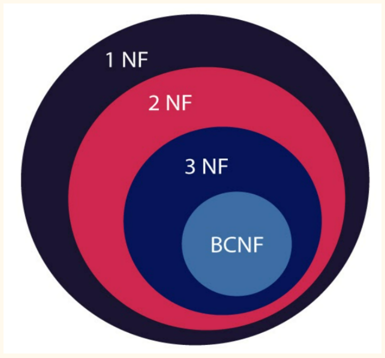
TESTING NORMALIZATION OF OUR TABLES :



Let's say that a table is in 1NF if it satisfies the below condition :-

" If there are no multivalued attributes , composite attributes , all columns have unique names and here the order in which data will be stored doesn't matter. "

Lets name this condition as 1NF condition.

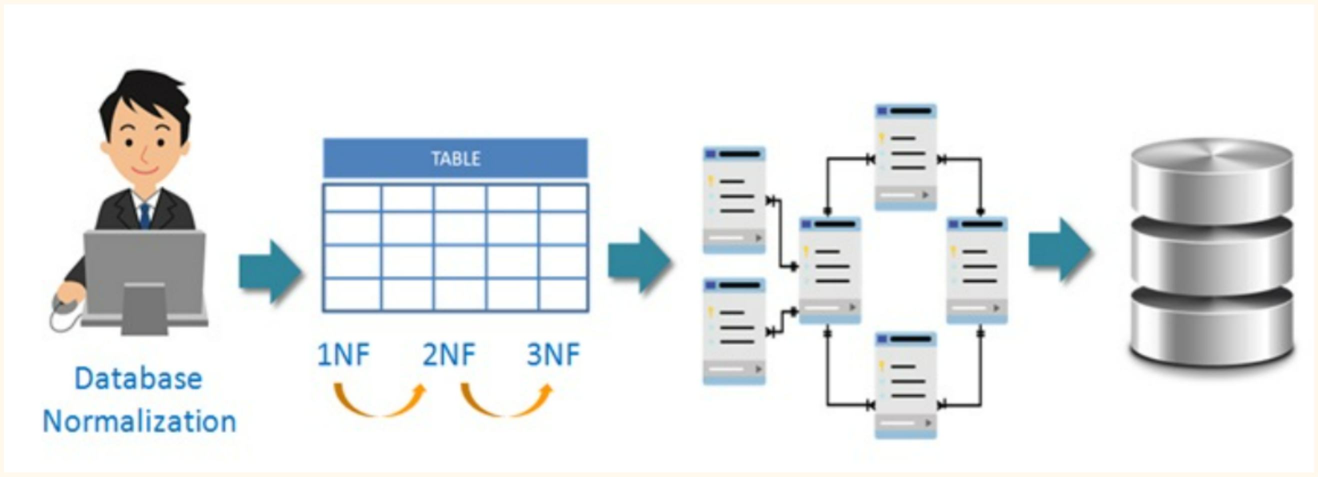


2 NF :- As we can see that all our candidate keys are single attributes . So , there exists no proper subset of candidate keys .

Therefore there is no chance of existence of partial dependency . Hence this table is in 2NF.

3 NF :- In our tables all functional dependencies will be from candidate key(prime attribute) to non prime attributes . Therefore there is no transitive dependency.

Hence this table is in 3rd normal form.



BCNF :- In all the above tables , only the superkeys are determining all other attributes.

Hence, we can say that the table is in BCNF .

Below Are The Tables :

Learner :

Attributes :

1. Learner_Id (integer, not null)
2. DOB (date)
3. Name (varchar, not null)
4. Gender (varchar)
5. Country (varchar)

SQL to create :

```
CREATE TABLE Learner(

  Learner_Id INT NOT NULL,

  DOB DATE,

  Name Varchar(20) NOT NULL,

  Gender VARCHAR(2),

  Country VARCHAR(20)  ,

  PRIMARY KEY (Learner_Id)

);
```

Category :

Attributes :

1. Category_Id (integer, not null)
2. Category_Name (varchar, not null)

SQL to create :

```
CREATE TABLE Category(

  Category_Id INT NOT NULL,

  Category_Name VARCHAR(20) NOT NULL,

  PRIMARY KEY (Category_Id)

);
```

Qualification:

Attributes :

1. qualification_id (integer, not null)
2. Graduation (varchar)
3. Post_Graduation (varchar)
4. Years_Of_Experience (integer)

SQL to create :

```
CREATE TABLE Qualification(

  qualification_id INT NOT NULL,

  Graduation VARCHAR(20),

  Post_Graduation VARCHAR(20),

  Years_Of_Experience INT,

  PRIMARY KEY (qualification_id)

);
```

Course:

Attributes :

1. Course_Id (integer, not null)
2. Name (varchar, not null)
3. Description (varchar)
4. Duration (integer, not null)
5. Price (integer, not null)
6. Category_Id (integer, not null)

SQL to create :

```
CREATE TABLE Course(
  Course_Id INT NOT NULL,
  Name VARCHAR(20) NOT NULL,
  Description VARCHAR(100) ,
  Duration INT NOT NULL,
  Price INT NOT NULL,
  Category_Id INT NOT NULL,
  PRIMARY KEY (Course_Id),
  FOREIGN KEY (Category_Id) REFERENCES Category(Category_Id)
);
```

Review:

Attributes :

1. Review_Id (integer, not null)
2. Rating (integer, not null)
3. Review_Title (varchar)
4. Description (varchar)
5. Learner_Id (integer, not null)
6. Course_Id (integer, not null)

SQL to create :

```
CREATE TABLE Review(

  Review_Id INT NOT NULL,

  Rating INT NOT NULL,

  Review_Title VARCHAR(20),

  Description VARCHAR(100),

  Learner_Id INT NOT NULL,

  Course_Id INT NOT NULL,

  PRIMARY KEY (Review_Id),

  FOREIGN KEY (Learner_Id) REFERENCES Learner(Learner_Id),

  FOREIGN KEY (Course_Id) REFERENCES Course(Course_Id)

);
```

Instructor:

Attributes :

1. Instructor_Id (integer, not null)
2. Name (varchar, not null)
3. Gender (varchar)
4. Country (varchar)
5. qualification_id (integer)

SQL to create :

```
CREATE TABLE Instructor(

  Instructor_Id INT NOT NULL,

  Name VARCHAR(20) NOT NULL,

  Gender VARCHAR(20),

  Country VARCHAR(20),

  qualification_id INT,

  PRIMARY KEY (Instructor_Id),

  FOREIGN KEY (qualification_id) REFERENCES Qualification(qualification_id)

);
```

Certificate:

Attributes :

1. Certificate_Id (integer, not null)
2. Certificate_Date (date, not null)
3. Learner_Id (integer, not null)
4. Course_Id (integer, not null)

SQL to create :

```
CREATE TABLE Certificate(
  Certificate_Id INT NOT NULL,
  Certificate_Date DATE NOT NULL,
  Learner_Id INT NOT NULL,
  Course_Id INT NOT NULL,
  PRIMARY KEY (Certificate_Id),
  FOREIGN KEY (Learner_Id) REFERENCES Learner(Learner_Id),
  FOREIGN KEY (Course_Id) REFERENCES Course(Course_Id)
);
```

Quiz:

Attributes :

1. quiz_id (integer, not null)
2. Quiz_number (integer)
3. Course_Id (integer, not null)
4. Instructor_Id (integer, not null)

SQL to create :

```
CREATE TABLE Quiz(

  quiz_id INT NOT NULL,

  Quiz_number INT,

  Course_Id INT NOT NULL,

  Instructor_Id INT NOT NULL,

  PRIMARY KEY (quiz_id),

  FOREIGN KEY (Course_Id) REFERENCES Course(Course_Id),

  FOREIGN KEY (Instructor_Id) REFERENCES Instructor(Instructor_Id)

);
```

Question:

Attributes :

1. question_id (integer, not null)
2. question (varchar(100), not null)
3. quiz_id (integer, not null)

SQL to create :

```
CREATE TABLE question(

  question_id INT NOT NULL,

  question VARCHAR(100) NOT NULL,

  quiz_id INT NOT NULL,

  PRIMARY KEY (question_id),

  FOREIGN KEY (quiz_id) REFERENCES Quiz(quiz_id)

);
```

Answer:

Attributes :

1. answer_id (integer, not null)
2. answer (varchar(10), not null)
3. question_id (integer, not null)

SQL to create :

```
CREATE TABLE answer(

  answer_id INT NOT NULL,

  answer VARCHAR(10) NOT NULL,

  question_id INT NOT NULL,

  PRIMARY KEY (answer_id),

  FOREIGN KEY (question_id) REFERENCES question(question_id)

);
```

Groupp:

Attributes :

1. group_id (integer, not null)
2. group_name (varchar(20), not null)
3. Instructor_Id (integer, not null)

SQL to create :

```
CREATE TABLE groupp(
  group_id INT NOT NULL,
  group_name VARCHAR(20) NOT NULL,
  Instructor_Id INT NOT NULL,
  PRIMARY KEY (group_id),
  FOREIGN KEY (Instructor_Id) REFERENCES Instructor(Instructor_Id)
);
```

Post:

Attributes :

1. id (integer, not null)
2. body (varchar(100))
3. group_id (integer, not null)

SQL to create :

```
CREATE TABLE post(
  id INT NOT NULL,
  body VARCHAR(100) ,
  group_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (group_id) REFERENCES groupp(group_id)
);
```

Learner_Course:

This table essentially has the primary keys of both the Learner and Course Table and is formed as a result of the many to many relationship between the two tables :

Attributes :

1. id (integer, not null)
2. body (varchar(100))
3. group_id (integer, not null)

SQL to create :

```
CREATE TABLE Learner_Course

(

  Learner_Id INT NOT NULL,

  Course_Id INT NOT NULL,

  PRIMARY KEY (Learner_Id, Course_Id),

  FOREIGN KEY (Learner_Id) REFERENCES Learner(Learner_Id),

  FOREIGN KEY (Course_Id) REFERENCES Course(Course_Id)

);
```

Instructor_Course:

Similar to the above table, this is formed as a result of the many-to-many relationship between the two tables :

Attributes :

1. Instructor_Id
2. Course_Id

SQL to create :

```
CREATE TABLE Instructor_Course
(
  Instructor_Id INT NOT NULL,
  Course_Id INT NOT NULL,
  PRIMARY KEY (Course_Id, Instructor_Id),
  FOREIGN KEY (Course_Id) REFERENCES Course(Course_Id),
  FOREIGN KEY (Instructor_Id) REFERENCES Instructor(Instructor_Id)
);
```

Solve:

Attributes:

1. grade
2. Learner_Id
3. quiz_id

SQL to create :

```
CREATE TABLE solve(

  grade VARCHAR(2),

  Learner_Id INT NOT NULL,

  quiz_id INT NOT NULL,

  PRIMARY KEY (Learner_Id, quiz_id),

  FOREIGN KEY (Learner_Id) REFERENCES Learner(Learner_Id),

  FOREIGN KEY (quiz_id) REFERENCES Quiz(quiz_id)

);
```

Joinn:

Attributes:

1. Learner_Id (integer, not null)
2. group_id (integer, not null)

SQL to create :

```
CREATE TABLE joinn(

  Learner_Id INT NOT NULL,

  group_id INT NOT NULL,

  PRIMARY KEY (Learner_Id, group_id),

  FOREIGN KEY (Learner_Id) REFERENCES Learner(Learner_Id),

  FOREIGN KEY (group_id) REFERENCES groupp(group_id)

);
```

Account:

Attributes:

1. Account_Id (integer, not null)
2. Account_Type (varchar(20), not null)
3. Learner_Id (integer)
4. Instructor_Id (integer)

SQL to create :

```
CREATE TABLE Account(
  Account_Id INT NOT NULL,
  Account_Type VARCHAR(20) NOT NULL,
  Learner_Id INT,
  Instructor_Id INT,
  PRIMARY KEY (Account_Id),
  FOREIGN KEY (Learner_Id) REFERENCES Learner(Learner_Id),
  FOREIGN KEY (Instructor_Id) REFERENCES Instructor(Instructor_Id)
);
```

Inserting Values in Tables :

We now insert data into all our tables.

The data must be inserted into the tables in the right topological order otherwise we will face the issues regarding integrity constraints.

Below is the sql to do the same :

```sql
--inserting values into Learner

insert into Learner values (1, '01-MAY-2003', 'Aman', 'M', 'India');

insert into Learner values (2, '11-JUNE-2003', 'Naman', 'M', 'India');

insert into Learner values (3, '07-JAN-2003', 'Raman', 'M', 'India');

insert into Learner values (4, '10-DEC-2003', 'Isha', 'F', 'India');

insert into Learner values (5, '04-SEPT-2003', 'Sneha', 'F', 'India');
```

```sql
--inserting values into Category

insert into Category values (1, 'Programming');

insert into Category values (2, 'Design');

insert into Category values (3, 'Business');

insert into Category values (4, 'Marketing');
```

```
--inserting values into Qualification

insert into Qualification values (1, 'B.Tech', 'M.Tech', 2);

insert into Qualification values (2, 'B.Com', 'M.Com', 2);

insert into Qualification values (3, 'B.A', 'M.A', 2);
```

```
--inserting values into Course

insert into Course values (1, 'Java', 'Java is a programming language', 2, 1000, 1);

insert into Course values (2, 'C++', 'C++ is a programming language', 3, 3000, 1);

insert into Course values (3, 'Stocks', 'Learn to invest', 4, 4000, 3);

insert into Course values (4, 'Marketing', 'Learn to market', 5, 5000, 4);

insert into Course values (5, 'Photoshop', 'Designing from scratch', 6, 6000, 2);

insert into Course values (6, 'Illustrator', 'Industry level Illustrator course', 7,
7000, 2);
```

```
--inserting values into Review

insert into Review values (1, 4, 'Good', 'Good course', 1, 1);

insert into Review values (2, 3, 'Average', 'Average course', 2, 2);

insert into Review values (3, 5, 'Excellent', 'Excellent course', 3, 3);

insert into Review values (4, 4, 'Good', 'Good course', 4, 4);

insert into Review values (5, 3, 'Average', 'Average course', 5, 5);
```

```sql
--inserting values into Instructor
insert into Instructor values(1, 'Ramakrishna', 'M', 'India', 2);
insert into Instructor values(2, 'Rajesh', 'M', 'India', 3);
insert into Instructor values(3, 'Priya', 'F', 'India', 3);
insert into Instructor values(4, 'Rahul', 'M', 'India', 1);
insert into Instructor values(5, 'Monica', 'F', 'India', 2);
```

```sql
--inserting values into Certificate
insert into Certificate values (1, '07-JUNE-2018', 1, 1);
insert into Certificate values (2, '11-NOV-2018', 2, 2);
insert into Certificate values (3, '06-OCT-2018', 3, 3);
insert into Certificate values (4, '08-DEC-2018', 4, 4);
```

```sql
--inserting values into Quiz
insert into Quiz values (1, 1, 1, 1);
insert into Quiz values (2, 2, 5, 2);
insert into Quiz values (3, 3, 6, 3);
insert into Quiz values (4, 4, 4, 4);
```

```sql
--inserting values into Question
insert into question values(1,'Is java object oriented?',1);
insert into question values(2,'Is photoshop a product of adobe',2);
insert into question values(3 ,'Is illustrator a product of adobe',3);
insert into question values(4,'Is marketing a good career option',4);
insert into question values(5,'Is java a product of microsoft',1);
```

```sql
--inserting values into Answer
insert into answer values(1,'yes',1);
insert into answer values(2,'no',2);
insert into answer values(3,'yes',3);
insert into answer values(4,'no',4);
insert into answer values(5,'yes',5);
```

```sql
--inserting values into Group
insert into groupp values(1,'Java Ki Padhai',1);
insert into groupp values(2,'C++ Ki Padhai',2);
insert into groupp values(3,'Stocks ka Group',3);
insert into groupp values(4,'Marketing ka Group',4);
insert into groupp values(5,'Photoshop pe baatein',5);
```

```
--inserting values into Post

insert into post values(1,'I have uploaded the codes on my github page. Check them
out',1);

insert into post values(2,'I will be rolling another course where I will continue
from where I left.',2);

insert into post values(3,'Steven will be your mentor, you may contact him
anytime',3);

insert into post values(4,'Follow my youtube channel, I will be sharing interview
tips regarding the subject.',4);
```

```
--inserting values into Learner_Course

insert into Learner_Course values(1,1);

insert into Learner_Course values(1,2);

insert into Learner_Course values(2,6);

insert into Learner_Course values(2,4);

insert into Learner_Course values(3,3);

insert into Learner_Course values(3,5);

insert into Learner_Course values(4,6);

insert into Learner_Course values(4,4);

insert into Learner_Course values(5,5);

insert into Learner_Course values(5,1);
```

```sql
--inserting values into Instructor_Course

insert into Instructor_Course values(1,1);

insert into Instructor_Course values(2,2);

insert into Instructor_Course values(3,3);

insert into Instructor_Course values(4,4);

insert into Instructor_Course values(5,5);

insert into Instructor_Course values(5,6);
```

```sql
--inserting values into Solve

insert into solve values('A',1,1);

insert into solve values('B',1,2);

insert into solve values('C',2,3);

insert into solve values('D',2,4);

insert into solve values('A',3,1);

insert into solve values('B',3,2);

insert into solve values('C',4,3);

insert into solve values('D',4,4);

insert into solve values('A',5,1);

insert into solve values('B',5,2);
```

```
--inserting values into Joinn

insert into joinn values(1,1);

insert into joinn values(1,2);

insert into joinn values(2,3);

insert into joinn values(2,4);

insert into joinn values(3,5);

insert into joinn values(3,1);

insert into joinn values(4,2);

insert into joinn values(4,3);

insert into joinn values(5,4);

insert into joinn values(5,5);
```

```
--inserting values into Account

insert into Account values(1,'Learner',1,NULL);

insert into Account values(2,'Learner',2,NULL);

insert into Account values(3,'Learner',3,NULL);

insert into Account values(4,'Learner',4,NULL);

insert into Account values(5,'Learner',5,NULL);

insert into Account values(6,'Instructor',NULL,1);

insert into Account values(7,'Instructor',NULL,2);

insert into Account values(8,'Instructor',NULL,3);

insert into Account values(9,'Instructor',NULL,4);

insert into Account values(10,'Instructor',NULL,5);
```

SQL Queries :

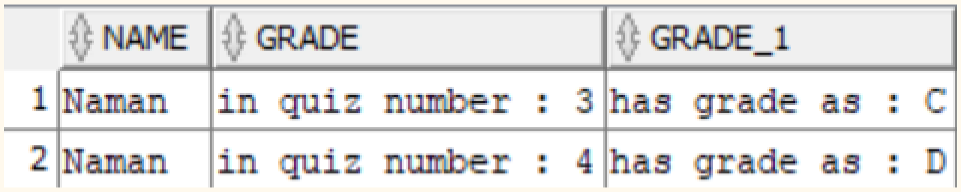
Below are some of the sql queries which might be often when such a database is put to use on an actual platform for students and instructors.

Q1. Find out the grades achieved by a particular learner in all the quizzes they have attempted till now.

SQL :

```
select name , concat('in quiz number : ' , quiz_number) as grade , concat( 'has grade
as : ' , grade) as Grade from learner

join solve on solve.learner_id = learner.learner_id

join quiz on quiz.quiz_id = solve.quiz_id

where learner.learner_id=2;
```

Output :



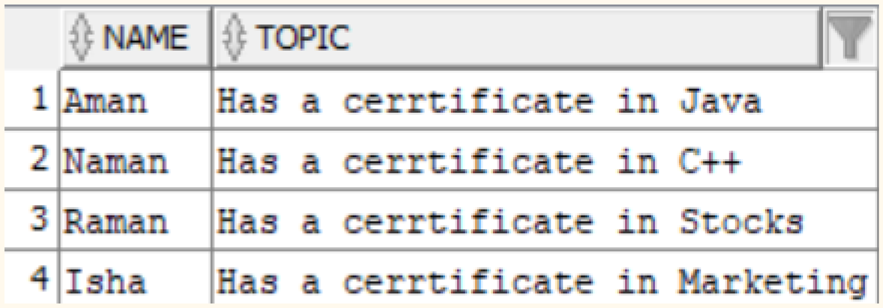
	NAME	GRADE	GRADE_1
1	Naman	in quiz number : 3	has grade as : C
2	Naman	in quiz number : 4	has grade as : D

Q2. Find out the certificates received by learners in all the courses they have enrolled in.

SQL :

```
select learner.name as Name , concat('Has a certificate in ' ,course.name) as Topic
from learner

join certificate on certificate.certificate_id = learner.learner_id

join course on course.course_id = certificate.course_id;
```

Output :



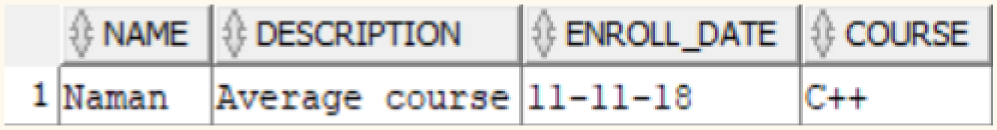
	NAME	TOPIC
1	Aman	Has a cerrtificate in Java
2	Naman	Has a cerrtificate in C++
3	Raman	Has a cerrtificate in Stocks
4	Isha	Has a cerrtificate in Marketing

Q3. Find out the reviews given on a particular course by students who have enrolled after 1-11-18 on that particular course.

SQL :

```
select learner.name , review.description , certificate_date as
enroll_date,Course.name as Course from Certificate

join learner on certificate.certificate_id = learner.learner_id

join course on course.course_id = certificate.course_id

join review on review.course_id = course.course_id

where certificate.Certificate_date >= '01-NOV-2018' AND Course.Course_id=2;
```

Output :



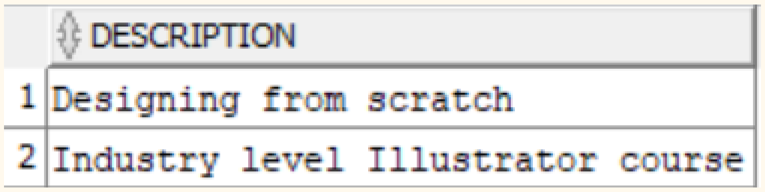
	NAME	DESCRIPTION	ENROLL_DATE	COURSE
1	Naman	Average course	11-11-18	C++

Q4. Find the questions in quiz for course having category as design.

SQL :

```
select course.description from category

join course on course.category_id = category.category_id

join quiz on quiz.course_id = course.course_id

join question on quiz.quiz_id = question.quiz_id

where category.category_name = 'Design';
```

Output :



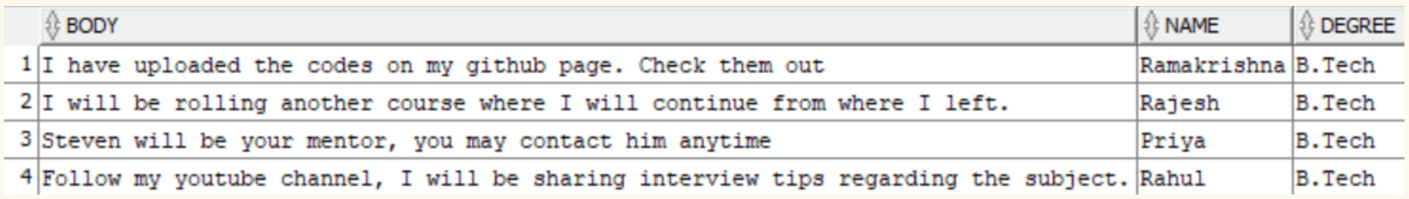
	DESCRIPTION
1	Designing from scratch
2	Industry level Illustrator course

Q5. Find the post uploaded by the instructors who have B.Tech as qualification and M.Tech as post graduation.

SQL :

```
select post.body , instructor.name , qualification.graduation as degree from post

join groupp on post.group_id = groupp.group_id

join instructor on groupp.instructor_id = instructor.instructor_id

join qualification on instructor.qualification_id = instructor.qualification_id

where qualification.graduation = 'B.Tech' and qualification.post_graduation =
'M.Tech';
```

Output :



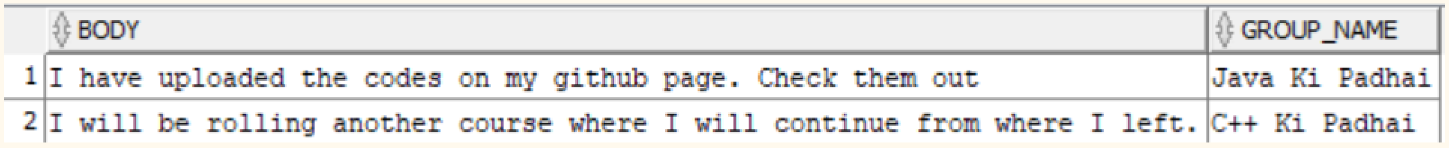
	BODY	NAME	DEGREE
1	I have uploaded the codes on my github page. Check them out	Ramakrishna	B.Tech
2	I will be rolling another course where I will continue from where I left.	Rajesh	B.Tech
3	Steven will be your mentor, you may contact him anytime	Priya	B.Tech
4	Follow my youtube channel, I will be sharing interview tips regarding the subject.	Rahul	B.Tech

Q6. Find the post and group which belong to the category of programming.

SQL :

```sql
select post.body , groupp.group_name from post

join groupp on post.group_id = groupp.group_id

join instructor on groupp.instructor_id = instructor.instructor_id

join Instructor_Course on Instructor_Course.instructor_id = instructor.instructor_id

join course on course.course_id = instructor_course.course_id

join category on course.category_id = category.category_id

where category.category_name = 'Programming';
```

Output :



	BODY	GROUP_NAME
1	I have uploaded the codes on my github page. Check them out	Java Ki Padhai
2	I will be rolling another course where I will continue from where I left.	C++ Ki Padhai

________________________X–X–X–X–X–X________________________