CPCS241 – Database I – Spring 2022 – Group Project

Dev Heroes

Consultation sessions reservation system Database

Contents

PART I: Analysis

1 Problem Definition and Data Requirements

1.1 Problem Description

Software developers, whether students or professionals, all face circumstances, in which they get stuck while working on some projects. Given that different developers have different views and experiences, collaborating on problems and sharing knowledge would be very beneficial. Although there are some very useful services for asking questions, like Stackoverflow[1], in many situations, direct and live communications help better convey the problem and allow for collaborative efforts to solve it. Dev Heroes enables software developers to connect with mentors and experts in different fields to share knowledge and collaborate on specific problems.

Via Dev Heroes, mentors can register and specify their fields of expertise, and their availability, and other developers can seek their help and reserve time slots to discuss and collaborate on problems or consult on technical issues.

1.2 Data Requirements

1) User entity

Each User has:

- ID: Unique id, primary key, must have 10 number, and is required.
- Name: A composite attribute that consists of the first name and last name each must have less than 50 characters and is required.
- Phone_number: Must have 10 numbers and is required.
- Email: Must be unique, have letters in English, digits, special characters, or all of them. Must be valid email. Is required.
- Start_date: Date of the user joining the application.
- Languages: A multivalued attribute that consists of languages the user can talk and understand.

Note: The user entity is a superclass with two sub classes: Mentor, and Mentee.

2) Mentor entity

Each Mentor has:

- Years_of_experience: Count the years of experience the mentor in the field. Must be positive number and is required.

[1] https://stackoverflow.com/

- Years_of_mentoring: Count the years of the mentor been in the application and is derived from the start date. Must be positive number and is required.
- Is_Active: Checks whether the mentor is currently online. Must be Boolean and is required.
- Availability: Check whether the mentor currently in session. Must be Boolean and is required.
- Meeting_method: Google meet – Zoom – Offline. Multivalued attribute. The mentor chooses the methods that he can accept as a meeting method with mentees. Is required.

3) Mentee entity

Each mentee has:

- Role: The mentee can be student, teacher, or employee. Is required.
- Score: Mentee's participation points in the application. Is required.

4) Field Entity

Each field has:

- ID: Primary key, must have 10 number, and is required.
- Topic: Multivalued like (IOS – Back-end - Front-end – Web development – Video game development – Mobile development). Is required.
- Description: Must have less than 1000 characters and is optional.

5) Discussion Room entity

Each discussion room has:

- ID: Primary key, must have 10 number, and is required.
- Name: must have less than 100 characters and is required.
- Description: Describe the problem. Must have less than 1000 characters and is optional.
- Start_date: date of starting the discussion room. Must be valid date and is required.
- Status: Completed, Canceled, Confirmed, Pending. Required and default is pending.
- Outcome: Fully resolved, partially resolved, Not resolved. Optional.

6) Blog entity

Each blog has:

- ID: Primary key, must have 10 number, and is required.
- Name: must have less than 50 characters and is required.
- Description: Describe the problem. Must have less than 1000 characters and is optional.

7) Post entity

- ID: Primary key, must have 10 number, and is required.
- Name: must have less than 100 characters and is required.
- Description: Describe the problem. Must have less than 1000 characters and is optional.

8) Session entity

Each session has:

- ID: Primary key, must have 10 number, and is required.
- Description: Must have less than 500 characters and is optional.
- Meeting_method: Google meet – Zoom – Offline. Is required.
- Start_time_stamp: Must be valid date. Is required.
- End_time_stamp: Must be valid date larger than start time stamp and is required.
- Status: Completed, Canceled, Confirmed, Pending. Required and default is pending.
- Outcome: Fully resolved, partially resolved, Not resolved. Optional.

9) Membership entity

Each membership has:

- ID: Primary key, must have 10 number, and is required.
- Tier: There are four tiers (gold – silver – bronze - standard). Each has their own features (Gold can have 5 hours session, silver can have 2 hours session bronze can have one hour session, free can have 40-minute session. Is required.
- Cost: Each tier has different costs: gold is 15 dollars a month, silver is 10 dollars a month, bronze is 5 dollar a month, and standard is free.
- Start_date: Start date of membership. Must be valid date and is required.
- End_date: It is a derivative value, which is the date of the last day for the membership. Must be valid date larger than start date and is required.

10) Comment entity

Each comment has:

- ID: Primary key, must have 10 number, and is required.
- AuthorID: ID of the user who wrote the comment.
- Content: must have less than 500 characters. Is required.
- Creation_date: Date of the creation of the comment.
- Update_date: Last date of update. Must be valid date and is required.
- Upvote: User can give upvote point to the comment.

11) Review entity

Each review has:

- ID: Partial key (Weak key), must have 10 number, and is required.
- Score: Must be from 1 to 5 and is required.
- Is_Anonymous: Boolean value whether the review is anonymous or not. Default value is false and is required.

1.3 Business Rules

1) User

- The database maintains all users' personal data. The system will auto-generate a unique ID [**ID**]. Each user has a composite attribute name that contain the first and last name [**Name**]. also, each user has email [**Email**] and phone number [**Phone_number**] to sign in the application. There is a multivalued attribute for the languages [**Languages**] that the user prefer / speaks. There is a [**Start_date**] attribute for the date of the user joining in the application. The user entity is a superclass of the subclasses: Mentor, and mentee.
- Each user has a [**Has**] relation with [**Field**] which is an entity that have data about what field the user is interested in / can teach.
- Each User can have many Fields, each field can have many users. There is no user without a field, but there could be a field without any user.

2) User / Mentor

- Each mentor has years of experience [**Years_of_experience**], which counts how many years they were in the field. There are also years of mentoring [**Years_of_mentoring**] which counts how many years the mentor has been in the application. It is a derivative attribute from the [**Start_date**] attribute in the superclass [**User**] and it must be positive number. Each mentor has [**Is_Active**] attribute, which checks whether the mentor is currently online. Must be Boolean value. Each mentor has [**Availability**] attribute, it checks whether the mentor currently in session. Must be Boolean value. Each mentor has [**Meeting_method**] multivalued attribute. The mentor chooses the methods that he can accept as a meeting method with mentees.
- Each mentor has a [**Manage**] relationship with [**Blog**], each mentor can manage and write in his own blog. Each mentor and mentee have a [**Has**] ternary relationship with session, they both holds a session.
- Each mentor can manage one blog, and each blog is managed by one mentor. There could be a mentor without a blog, but there is no blog without a mentor.

3) User / Mentee

- Each mentee has a [**Role**] attribute, means the mentee can be student, teacher, or employee. Each mentee has [**Score**] attribute which counts mentee's participation points in the application.
- Each mentor has [**own**] relation with membership. Each mentor has [**own**] relation with discussion room that can start a conversation in.
- Each mentee owns many discussion rooms, but each discussion room is owned by one mentee. Every mentee can have zero or more discussion rooms, but every discussion room must have at least one mentee.

- Each mentee owns one membership, but each membership is owned by many mentees. Every mentee must have membership, but every membership can have zero or more mentees.

4) Field

- Each field entity has unique [**ID**] that the system will auto-generate. Each field will have a [**Topic**] multivalued attribute, such as: IOS – Back-end - Front-end – Web development – Video game development – Mobile development. Each field has a description that must have less than 1000 characters, nullable, and is optional. Each field has a [**Description**] attribute that describes the field.
- Each field has a [**Has**] relation with blog, session, post, and discussion room.
- Each field has many blogs, and every blog has many fields. A field could have zero blogs, but every blog must have at least one field.
- Each field has many sessions, and every session has many fields. A field could have zero sessions. but every blog must have at least one session.
- Each field has many posts, and every post has many fields. A field could have zero posts. but every post must have at least one session.
- Each field has many discussion rooms, and every discussion room has many fields. A field could have zero discussion rooms, but every discussion rooms must have at least one field.

5) Discussion room

- Each discussion room entity has unique [**ID**] that the system will auto-generate. Each discussion room must have a name [**Name**] that contain less than 100 characters. There could be a description that describe the problem [**Description**]. Each discussion room has a [**Start_date**] attribute is the date of starting the discussion room. Each discussion room has a [**Status**] attribute that contain one of these values: Completed, Canceled, Confirmed, Pending. Each discussion room has a [**Outcome**] attribute that contain one of these values: Fully resolved, partially resolved, Not resolved.
- Each discussion room has a [**Own**] relation with mentee, [**Has**] relation with field, and [**Has**] relation with comment.
- Each discussion room is owned by one mentee, but every mentee owns many discussion rooms. A discussion room can’t have zero mentee, but a mentee could have zero discussion rooms.
- Each discussion room has many fields, and a field could have many discussion rooms. A discussion room can be without a field, but there could be a field without a discussion room.
- Each discussion room have many comments, but a comment could have one discussion room. A discussion room can have zero comments, but a comment can’t have zero discussion rooms.

6) Blog

- Each blog entity has unique [**ID**] that the system will auto-generate. Each blog must have a name [**Name**] that contain less than 50 characters. There could be a description that describe the problem [**Description**].
- Each blog has a [**Manage**] relation with mentor, [**Has**] relation with post, and [**Has**] relation with post.
- Each blog is managed by one mentor, and each mentor could manage one blog. A blog can't have zero mentors, but a mentor could have zero blogs.
- Each blog has many posts, but a post can have one blog. A blog can have zero or more posts, but a post can't have zero blogs.

7) Post

- Each post entity has unique [**ID**] that the system will auto-generate. Each post must have a name [**Name**] that contain less than 100 characters. There could be a description that describe the problem [**Description**].
- Each post has a [**Has**] relation with comment, blog, and field.
- Each post has many comments, but a comment could have one post. A post can have zero comments, but a comment can't have zero posts.
- Each Post has one blog, but a blog can have many posts. A post can't have zero blogs, but a blog can have zero posts.
- Each post has many fields, and a field could have many posts. There is no post without a field, but there could be a field without a post.

8) Session

- Each session entity has unique [**ID**] that the system will auto-generate. There could be a description that describe the problem [**Description**]. Each session has [**Meeting_method**] which is the method for the session. [**Start_time_stamp**] attribute is the time and date for the start of the session. And [**End_time_stamp**] is the time and date for ending of the session. Must be valid date larger than start time stamp and is required. Every session has a [**Status**] attribute that contain one of these values: Completed, Canceled, Confirmed, Pending. Every session has a [**Outcome**] attribute that contain one of these values: Fully resolved, partially resolved, Not resolved.
- Each session has a [**Has**] ternary relation with both mentor and mentee, [**Has**] relation with review, and [**Has**] relation with field.
- Each session must have one mentee and one mentor, but a mentor and a mentee must have one session. There can't be a session without a mentor and a mentee, but there can be a mentor and mentee without a session.

- Each session has one review, and a review has one session. There could be a session without review but can't be a review without a session.
- Each session can have many fields, and a field can have many sessions. There can't be a session with zero fields, but there could be a field without a session.

9) Membership

- Each membership entity has unique [**ID**] that the system will auto-generate. Each membership has a [**Tier**] list, there are four tiers (gold – silver – bronze - standard). Each has their own features (Gold can have 5 hours session, silver can have 2 hours session bronze can have one hour session, free can have 40-minute session. Each membership has a [**Cost**] that each tier has different costs: gold is 15 dollars a month, silver is 10 dollars a month, bronze is 5 dollar a month, and standard is free. Each membership has [**Start_date**], which is the start date of membership. There is also a [**End_date**], which is a derivative value from [**Start_date**]. The date of the last day for the membership.
- Each membership has a [**Own**] relation with the mentee.
- Each membership is owned by many mentees, but each mentee owns one membership. There could be a membership without a mentee but can't be a mentee without a membership.

10) Comment

- Each comment entity has unique [**ID**] that the system will auto-generate. Each comment has the ID of the author [**AuthorID**] who wrote the comment. Each comment contain a [**Content**] attribute. [**Creation_date**] is the date of the creation of the comment. [**Update_date**] is the date of the last update of the comment. [**Upvote**] is how many upvote did the comment get, which is a system similar to "likes" that the user can give to the comment.
- Each comment has a [**Has**] relation with discussion room, review, and post.
- Each comment has one discussion room, but each discussion room could have many comments. There can't be a comment with zero discussion room, but there can be a discussion room with zero comments.
- Each comment has one review, and each review could have one comment. There can't be a comment with zero review, but there can be a review with zero comments.
- Each comment has one post, but each post can have many comments. A comment can't have zero posts, but a post can have zero comments.

11) Review

- Each review entity has partial [**ID**] that the system will auto-generate, which is a weak key. Because the review entity depends on the session entity. [**Score**] is a scoring system for the review that the mentee writes for the session, it must be from 1 to 5. [**Is_Anonymous**] is a Boolean value for whether the review is anonymous or not.
- Each review has a [**Has**] relation with session, and comment.

- Each review has one session and each session have one review. A review can’t be with zero sessions, but there can be a session without a review.
- Each review has one comment, and each comment has one review. A review could have zero comments, but a comment can’t have zero review.

1.4 Intended Output of the system

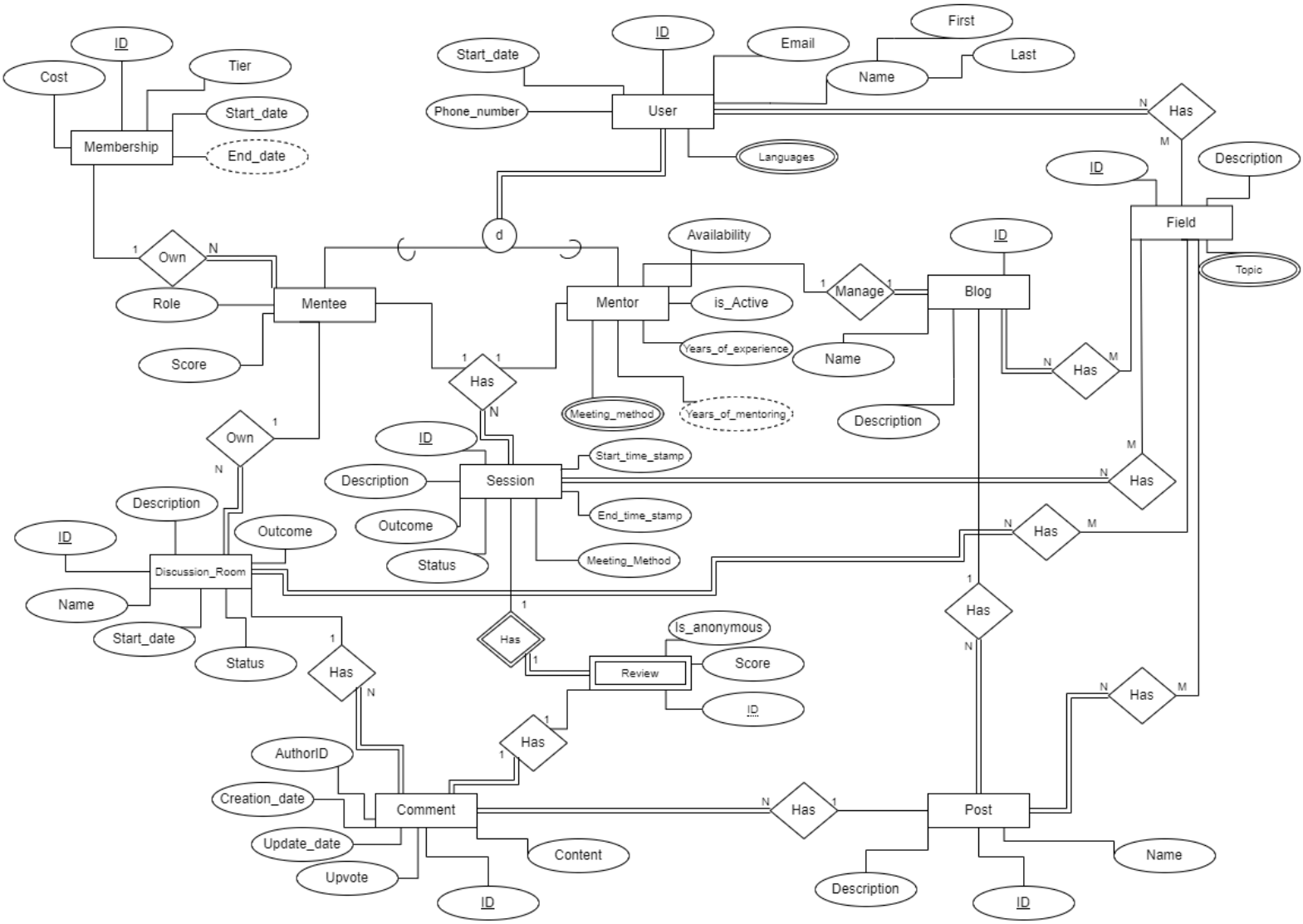
❖ Queries and Outputs

1. Average session rating for mentor.
2. Percentage of status for sessions.
3. Get pending sessions for mentors.
4. Get sessions for today (Mentor / Mentee)
5. Get all reviews for a mentor
6. Top mentors by number of sessions
7. Get count of confirmed sessions (Mentor / Mentee)
8. Get count of reviews (Mentor / Mentee)
9. Percentage of outcomes per session
10. Average session duration
11. Percentage fully resolved per group of years of experience (The groups: less than five years, from five to ten years, more than 10 years)
12. The period of being a mentor.
13. The mentor who has the highest period in being a mentor.
14. The blog with must comments.
15. Total profit from the membership.
16. Get all ended sessions of today.
17. Get all comments

PART II: DB DEISGN

2 ER Diagram Design

2.1 ER Diagram



2.2 Design of Business Rules

Business Rule	Design Decisions	Justification (if any)
The business rule in an ER relation		
User must be a mentee or mentor	Superclass/subclass relationship with disjoint and complete constraints	• Since each user must be either mentee or mentor (can't be both), it's disjointed and total participation.
A session has a field	N:M binary relationship between session and field.	• A session can hold multiple fields, and the same fields could be talked about in more than one session. • There is no session without specific fields. So, it is total participation from a session side. • There can be a field that has not been covered by any session. So, it's partial participation on a field side.
Each session must have one mentee and one mentor	N:1:1 ternary relationship between session, mentee, and mentor.	• A mentee can join multiple sessions, but a session is held for only one mentee. • A mentor has multiple sessions, but a session is held by only one mentor. • The mentee will choose one mentor to take the session with, and each mentor will hold the session for only one mentee.

		• The session will not hold if there is no mentee and mentor. So, it is total participation on both mentee and mentor sides. • A mentee has the choice of joining the session or rejecting, and the mentor has also the choice of holding the session or rejecting it, hence its partial participation on both the mentee and mentor sides.
Each session can have a review	1:1 weak binary relationship between session and review	• each session can have one review and each review belongs to only one session. • If a session does not exist, the review will not exist as well. • Since the review depends on a session, it is displayed in total participation. • The mentee in the session has the option to fill out the review or not. So, it's partial participation from a session side.
Each review can have a comment	1:1 binary relationship between review and comment.	• Each review can have one comment, and each comment belongs to only one review • The mentee has the option to write a comment or not, so it is

		partial participation from a review side. • The mentee will not be able to write a comment until the review appears, so it is total participation from a comment side.
Each mentor manages a blog	1:1 binary relationship between mentor and blog.	• A mentor can have one blog that can publish different topics on it, and each blog belongs to only one mentor • the mentor has the option to manage a blog or not, so it is partial participation from a mentoring side. • each blog must have one mentor to manage it, so it is total participation from a blog side.
Each blog can have multiple fields	N:M binary relationship between blog and field.	• A mentor can choose more than one field in his blog, and the same field can be on more than one blog • Each blog must have one or more fields, so it is total participation from the blog side. • There can be a field that has not been chosen by any blog, so it is partial participation from a field side.

Each blog has a post	1:N binary relationship between blog and post.	• Each blog contains zero or more posts, but each post belongs to only one blog. • Each blog continues zero or more posts so, it is partial participate on from a blog side. • Each post must belong to only one blog, so it is total participation from the blog side.
Each post can have multiple fields	N:M binary relationship between post and field.	• Each post must talk about one or more specific fields, and the same field can be on more than one blog • Each post must talk about one or more fields, so it is total participation from the post side. • There can be a field that has not been chosen by any post, so it is partial participation from a field side.
Each user has a field	N:M binary relationship between user and field.	• The mentor must choose the fields that can teach it, and the mentee must also choose the fields that interested in. The same fields can take by many users. • Each user (mentee and mentor) must choose one or more fields, so it is total participation from the user side.

		• There can be a field that has not been chosen by any user, so it is partial participation from a field side.
Each post can have multiple comments.	1:N binary relationship between b post and comment.	• each comment belongs to only one post, and each post may contain zero or one comment. • It is not necessary for each post to contain comments, so it is partial participation from a post side. • each comment written must be specific to one post, so it is total participation from the comment side.
Each mentee own membership	N:1 binary relationship between mentee and membership.	• Each mentee, upon registration, must choose one membership, and the same membership can be chosen by more than one mentee • Each mentee must choose one membership, so it is total participation from the mentee side. • There may be memberships that are not selected by any mentee, so it is partial participation from the membership side.

Each mentee owns a discussion room	1:N binary relationship between mentee and discussion room.	• Each mentee has the possibility to own discussion rooms to discuss any topic or problem with other mentees, and each discussion room has only one owner. • There may be a mentee who has not previously opened a discussion room, so it is partial participation from a mentee side. • Every discussion room should have an owner, so it is total participation from the discussion room side.
Each discussion room has a comment	1:N binary relationship between discussion room and comment.	• In each discussion room, other users can write a comment in response to the problem or topic at hand, and each comment belongs to only one post. • Some discussion rooms will not have comments, so it is partial participation from a discussion room side. • Each comment written by the users must belong to one discussion room, so it is total participation from the comment side.

Each discussion room has a field	N:M binary relationship between discussion room and field.	• A discussion room can hold multiple fields, and the same fields could be talked about in more than one discussion room. • There is no discussion room without specific fields. So, it is total participation from a discussion room side. • There can be a field that has not been covered by any discussion room. So, it's partial participation on a field side.
The business rule in a relation schema and queries		
Each user {mentor, mentee} has a • **Id** • **Name** • **Phone number** • **Email** • **Start date**	• User's Id **[ID]** is not nullable. • User's name **[Name]** is not nullable. • User's Phone number **[Phone_number]** is not nullable. • User's Email **[Email]** is not nullable • User's start date **[Start_Date]** is not nullable	• The database maintains users' information, and the system will auto-generate 10 unique numbers **[ID]** for every user, so it is not nullable. • Every user has a name **[Name]** continued two-part (first and last name), 10-digit phone number **[Phone_number]**, and email **[Email]**, so they cannot be nullable. • Each user has a start date **[Start_Date]** that is automatically calculated upon registration.
Each mentor has also: • **Years of experience** • **Is active** • **Availability** • **Meeting method**	• Mentor experience years **[Years_of_experience]** is not nullable. • Is the mentor active **[isActive]** is not nullable.	• Each mentor should have a positive number of years of experience **[Years_of_experience].**

• **Years of Mentoring**	• is the mentor available **[Availability]** is not nullable. • Meeting method **[Metting_method]** is not nullable • mentoring years **[Years_of_Mentoring]** is not nullable	• Each mentor has isActive Boolean **[is_Active]** attribute which tells the mentee whether the mentor is online or offline, the availability **[Availability]** is used to show whether the mentor is currently busy in a session or not. • About the meeting method **[Metting_method]**, every mentor will choose one type out of all the multivalued. • The mentoring years of every mentor will be related to the start date **[Years_of_mentoring]**, in another word, the mentoring years can be derivable from the start date.
Each mentee has also: • **Role** • **Score**	• Role **[Role]** is not nullable. • Score **[Score]** of the mentee is not nullable.	• Each mentee will choose their role **[Role]** upon registration. The mentee can be a student, teacher, or employee. • the mentee will have participation points **[Score]** in the application.
Each session has: • **ID** • **Meeting method** • **Description** • **Start timestamp**	• Session-id **[ID]** is not nullable. • meeting method **[metting_method]** is not nullable.	• The system will auto-generate 10 unique numbers **[ID]** for every session, so it is not nullable.

• **end timestamp** • **Outcomes** • **Statues**	• session description **[Description]** is nullable. • Session Start timestamp **[Start_time_stamp]** is not nullable • Session End timestamp **[End_time_stamp]** is not nullable • Outcomes of the session **[Outcomes]** is nullable • Status of the session **[Status]** is not nullable	• Each session has a specific meeting method **[metting_method]** in which the mentor and mentee can enter the session. • The description **[Description]** in the session is not required. The mentor can write for example the goals and outputs in less than 500 characters. • The start timestamp **[Start_time_stamp]** is determined by the mentor • end timestamp. **[End_time_stamp]** will be determined auto-according to the selected membership. • The mentor can write the outcomes of the session **[Outcomes]** after the session is ended. • The statue of the session **[Status]** is determined by the both the mentor and mentee. They decide whether the problem was fully resolved, partially resolved, or not resolved.
Each field has: • **ID** • **Topic** • **Description**	• Topic id **[ID]** is not nullable. • Topic Category **[Topic]** is not nullable. • Topic description **[Description]** is nullable.	• For each field, the system will auto-generate a 10 unique id **[ID]**. • The field also should have a Topic **[Topic]** that is multivalued.

		• The description **[Description]** in the field is not required it can be 1000 characters or less.
Each Review has: • **ID** • **Score** • **is anonymous**	• Review id **[ID]** is not nullable. • The score **[Score]** is not nullable. • is anonymous **[is_anonymous]** is not nullable.	• For each Review, the system will auto-generate a 10 unique id **[ID]**. • The mentee should evaluate the session and give a score **[Score]** from 1 to 5. • The mentee can choose whether his name appears when sending the review or not **[is_anonymous]**. the default value is false, which means the comment is not anonymous and the username will appear.
Each membership has: • **ID** • **Tier** • **Cost** • **Start_date** • **End_date**	• membership id **[ID]** is not nullable. • Tier **[Tier]** id not nullable • Cost **[Cost]** is not nullable • Start_date **[Start_date]** is not nullable • End_date **[End_date]** is not nullable	• For each membership, the system will auto-generate a 10 unique id **[ID]**. • There are 4 tier **[Tier]** and each with their own feature. • Each tier had different cost **[Cost].** • In each registration, the system will auto-generate the start date **[Start_date].** • end date **[End_date]** will be derivable from the start date.
Each discussion room has: • **ID** • **Description**	• discussion room id **[ID]** is not nullable.	• For each discussion room, the system will

• **Name** • **Outcome** • **Statue** • **Start_date**	• discussion room description **[Description]** is nullable. • discussion room **[Name]** is not nullable • Outcomes of the discussion room **[Outcomes]** • is nullable • Status of the discussion room **[Status]** is not nullable • **[Start_date]** is not nullable	auto-generate a 10 unique id **[ID]**. • The description **[Description]** in the discussion room is not required. The mentee can write for example a description of the problem or topic in less than 1000 characters. • The mentee can write the outcomes of the session **[Outcomes].** • The mentee should choose a name **[Name]** for the session that describes the problem he is facing in less than 100 characters. • The status of the session **[Status]** is determined by the mentee. it decides whether the problem was fully resolved, partially resolved, or not resolved. • The system will automatically generate a start date **[Start_date]** for the discussion room
Each blog has: • **ID** • **Description** • **Name**	• blog id **[ID]** is not nullable. • blog description **[Description]** is nullable. • blog **[Name]** is not nullable	• For each blog, the system will auto-generate a 10 unique id **[ID]**. • The description **[Description]** in the blog is not required. The mentor can write a description of the blog in less than 500 characters.

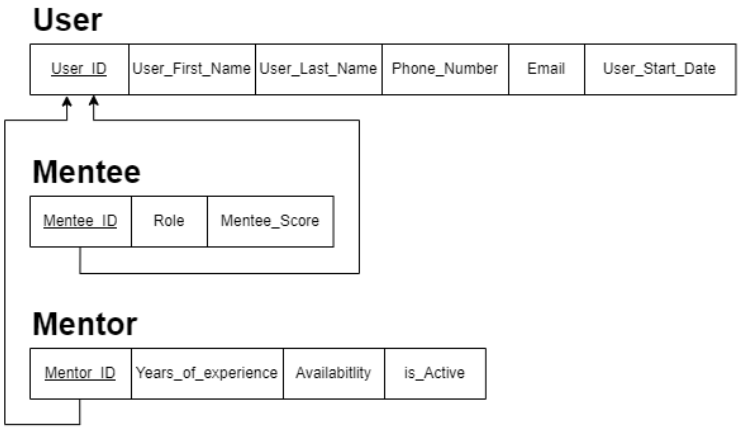
		• The mentor should choose a name **[Name]** for the blog that describes her\his own blog in less than 50 characters.
Each post has: • **ID** • **Description** • **Name**	• post id **[ID]** is not nullable. • post description **[Description]** is nullable. • post **[Name]** is not nullable	• For each post, the system will auto-generate a 10 unique id **[ID]**. • The description **[Description]** in the post is not required. The mentor can write a description for each post in less than 1000 characters. • The mentor should choose a name **[Name]** for each post that will posted in less than 100 characters.
Each comment has: • **ID** • **Author id** • **Content** • **Creation date** • **Update date** • **Upvote**	• comment id **[ID]** is not nullable. • author id **[Author id]** • is not nullable. • content **[content]** is not nullable • Creation date. **[Creation_date]** is not nullable. • Update date **[Update_date]** is not nullable. • Upvote **[upvote]** is nullable.	• For each comment, the system will auto-generate a 10 unique id **[ID]**. • ID **[Author id]** of the user who wrote the comment. • The system will auto-generate the date the comment was created **[Creation_date],** and the date it was modified **[Update_date].** • User can give upvote**[upvote]** point to the comment.

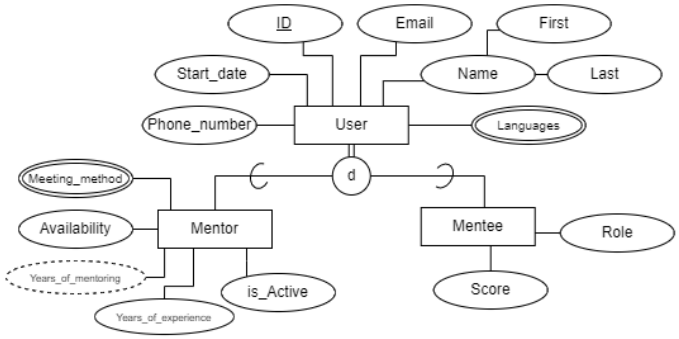
3 ER-to-logical Schema Mapping

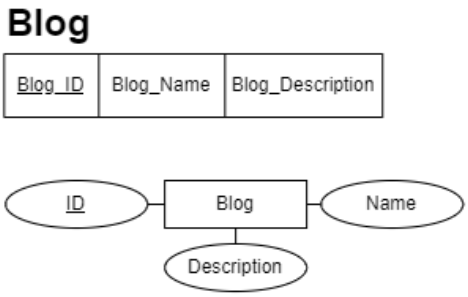
3.1 Mapping of Regular Entity Types

- We used the superclass and subclass to avoid redundancy, save the designer's time and make the ER diagram more readable. Each mentee and mentor have similar attributes; therefore, we added the user entity to be the superclass. It's a disjoint superclass relationship because each User can be either a mentor or mentee, not both.

- The derived attribute [**Years_of_mentoring**] is not included now it will be included later in the implementation phase.

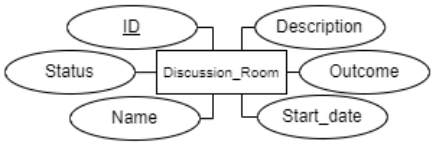






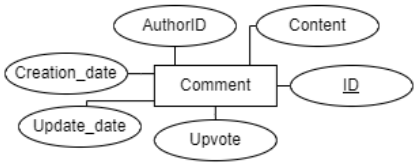
Discussion_Room

Disscussion_Room_ID	Disscussion_Room_Name	Disscussion_Room_Description	Disscussion_Room_Start_date	Status	Outcome



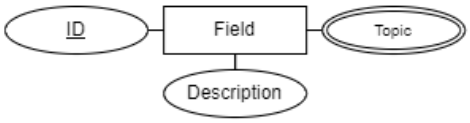
Comment

Comment_ID	AuthorID	Content	Creation_date	Update_date	Upvote



Field

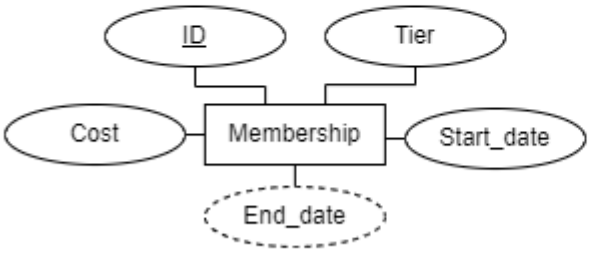
Field_ID	Field_Description



- The derived attribute [**End_Date**] is not included now it will be included later in the implementation phase.

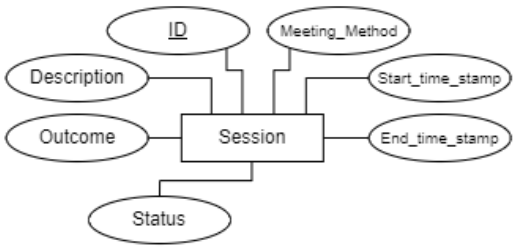
Membership

Membership_ID	Tier	Cost	Membership_Start_date



Session

Session_ID	Session_Description	Session_Meeting_Method	Session_Outcome	Start_time_stamp	End_time_stamp	Session_Status

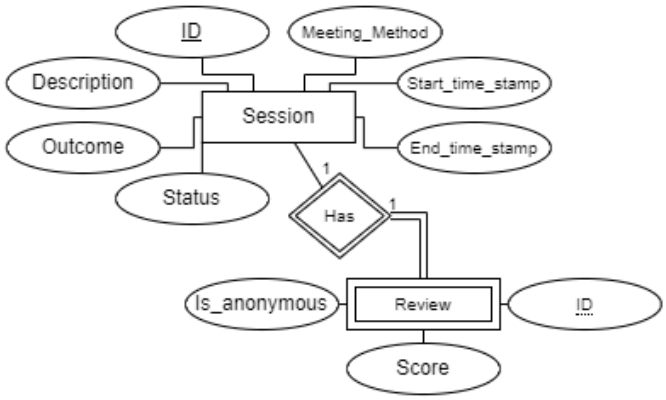


Post

Post_ID	Post_Name	Post_Description

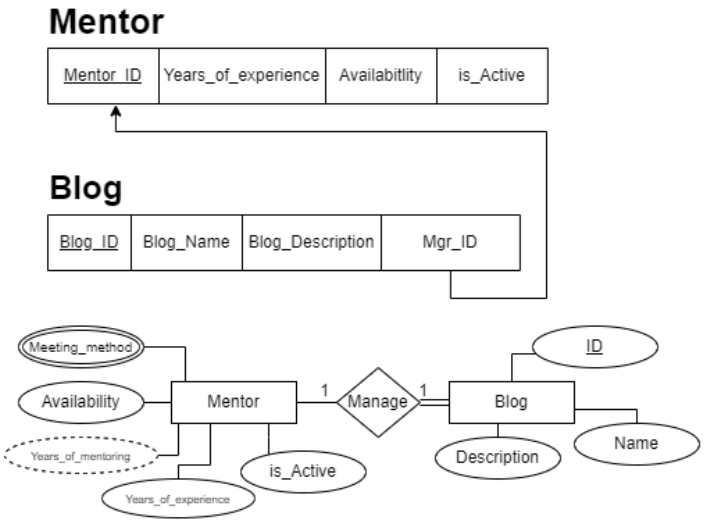
3.2 Mapping of Weak Entity Types

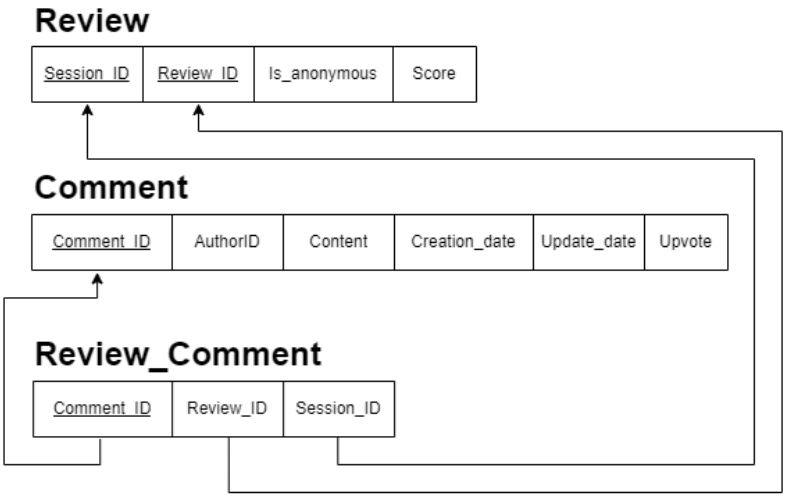
have one weak entity which is [**Review**] because it is dependent on another entity which is [**Session**]. there is no review without session. [**Review**] has a partial key which is [**ID**], and it is 1:1 binary relationships and total from one side. Each review has one session and each session have one review. A review can't be with zero sessions, but there can be a session without a review. the primary key will consist of identifying entity as a foreign key and a partial key for the weak entity.

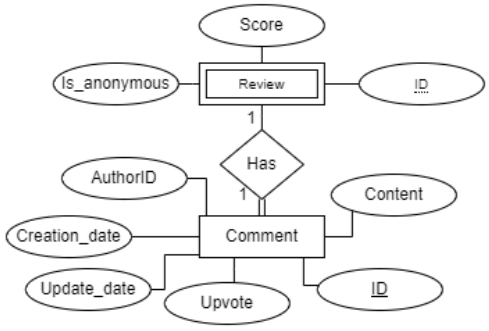


3.3 Mapping of Binary 1-1 Relationship Types

- One [**Mentor**] can write one blog.
- Each [**Review**] has one comment to be written.



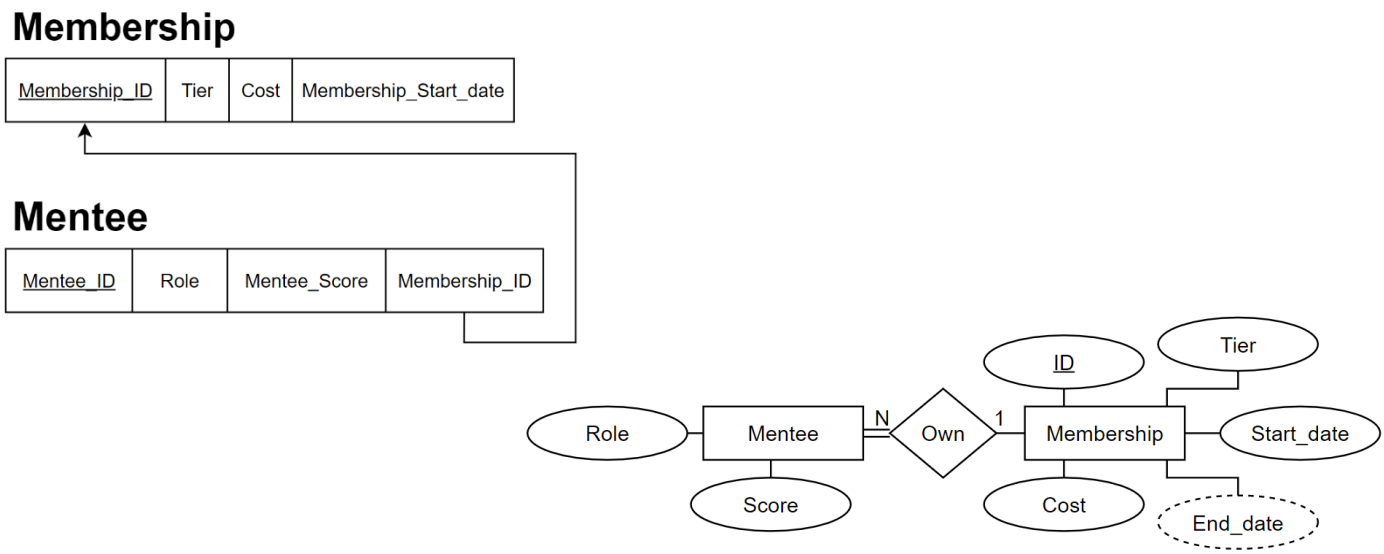




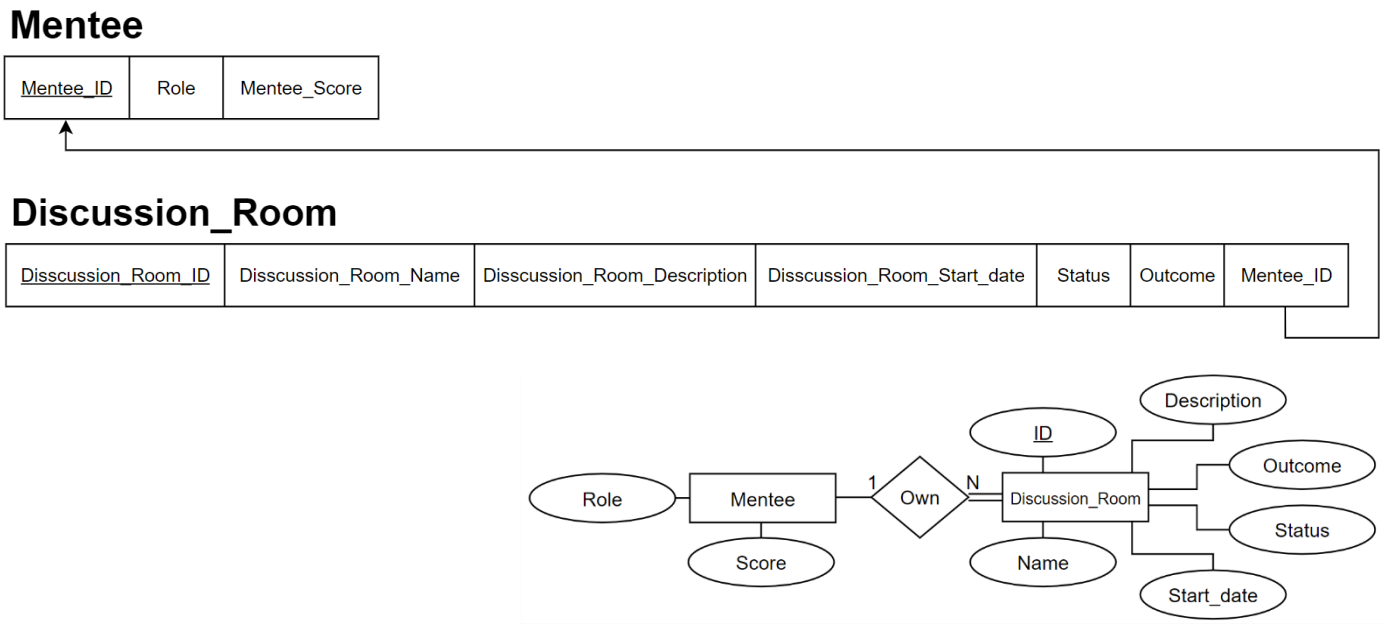
3.4 Mapping of Binary 1-N Relationship Types

There is more than one approach to map binary 1-N relationship types, one of them is the "foreign key approach", where we include the Primary Key of the entity in the 1 side as a Foreign Key in the entity in the N side. Another approach is the "cross reference approach", where we create a new relation, add the Primary Keys of the two relationships in the new relation as a Foreign Keys, the Primary Key of this new relation is the same as the Primary Key of the entity in the N side.

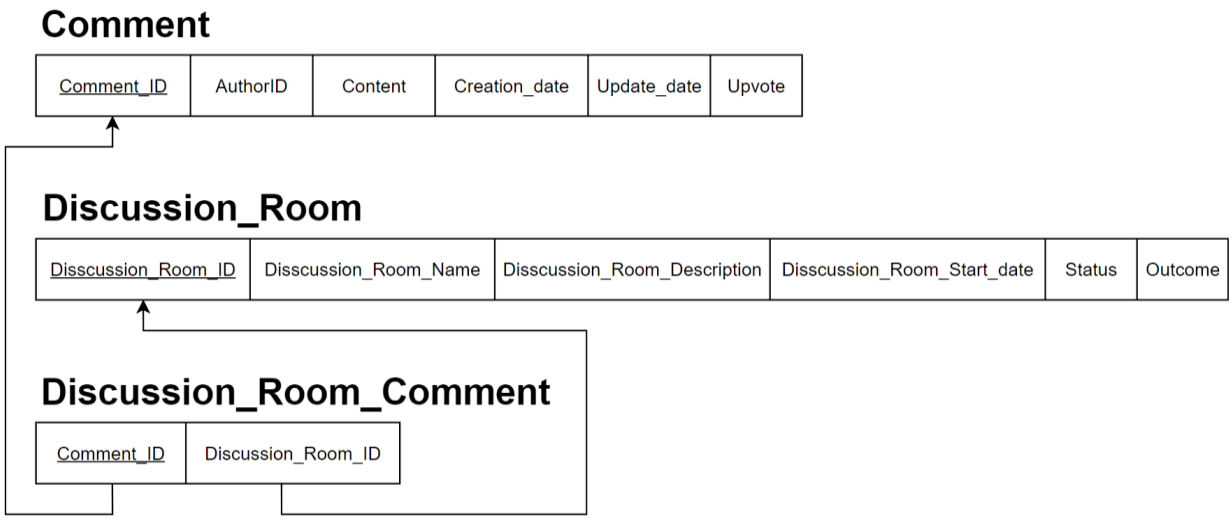
We used the "foreign key approach" in this relationship, we included the Primary Key of [**Membership**] in [**Mentee**] as a Foreign Key.

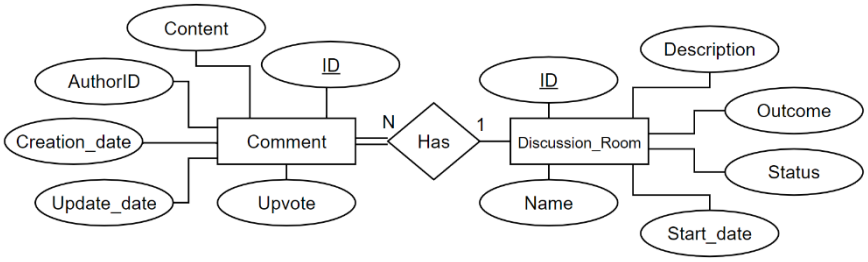


We used the "foreign key approach" in this relationship, we included the Primary Key of [**Mentee**] in [**Discussion_Room**] as a Foreign Key.

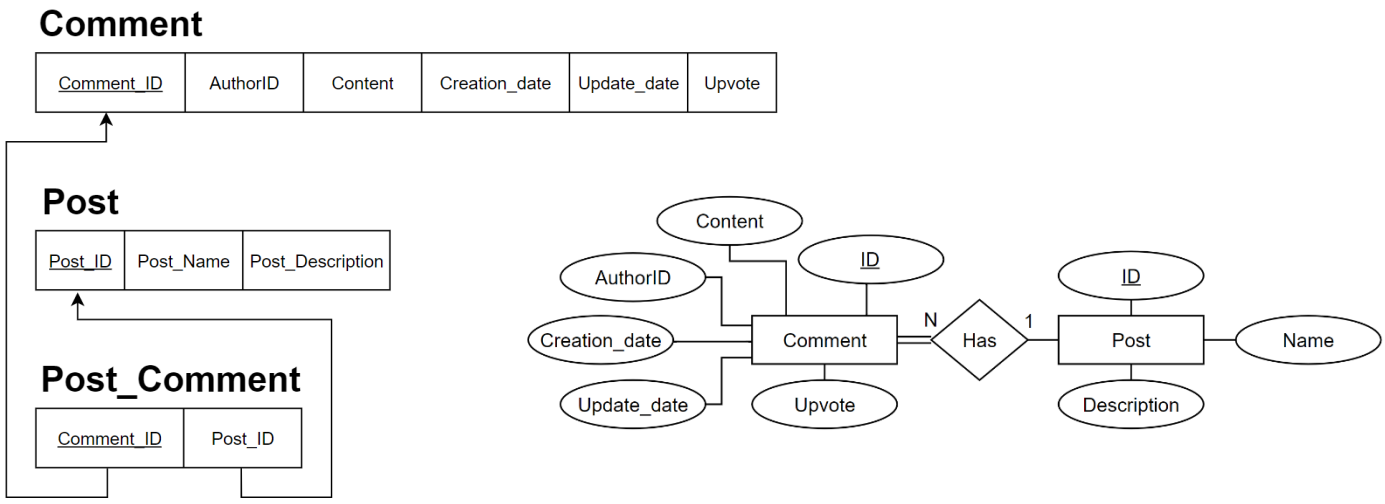


We used the “cross reference approach” in this relationship, we created a new relation, included the Primary Key of [**Comment**] and [**Discussion_Room**] in this relation as a Foreign Keys. We used this approach since [**Comment**] has more than one [**Has**] relation, and to make it more readable and understandable, we create a new relation for [**Discussion_Room_Comment**].

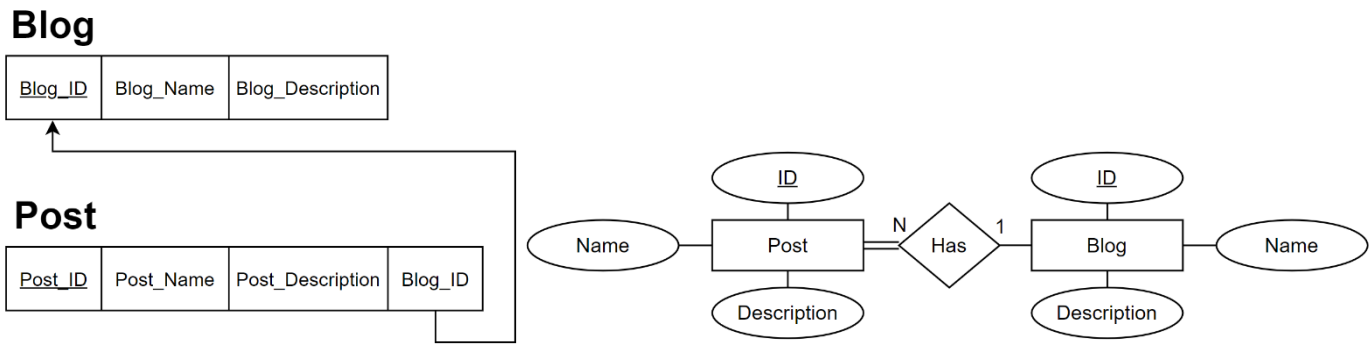




We used the “cross reference approach” in this relationship, we created a new relation, included the Primary Key of [**Comment**] and [**Post**] in this relation as a Foreign Keys. We used this approach since [**Comment**] has more than one [**Has**] relation, and to make it more readable and understandable, we create a new relation for [**Post_Comment**].

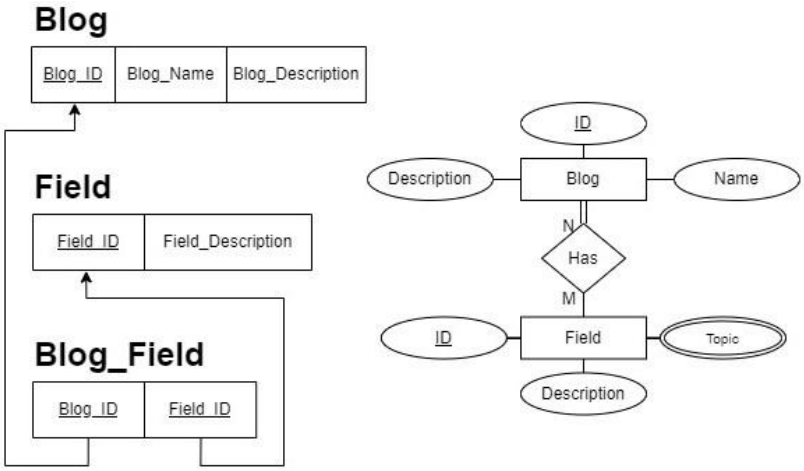


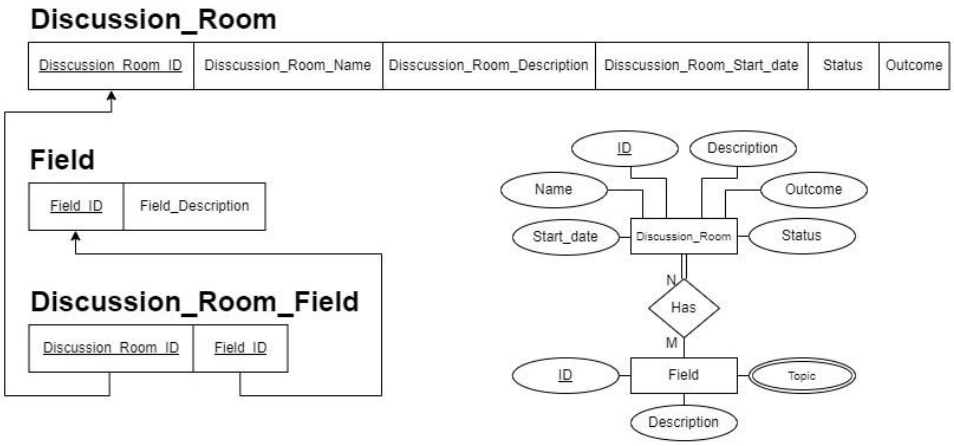
We used the “foreign key approach” in this relationship, we included the Primary Key of [**Blog**] in [**Post**] as a Foreign Key.

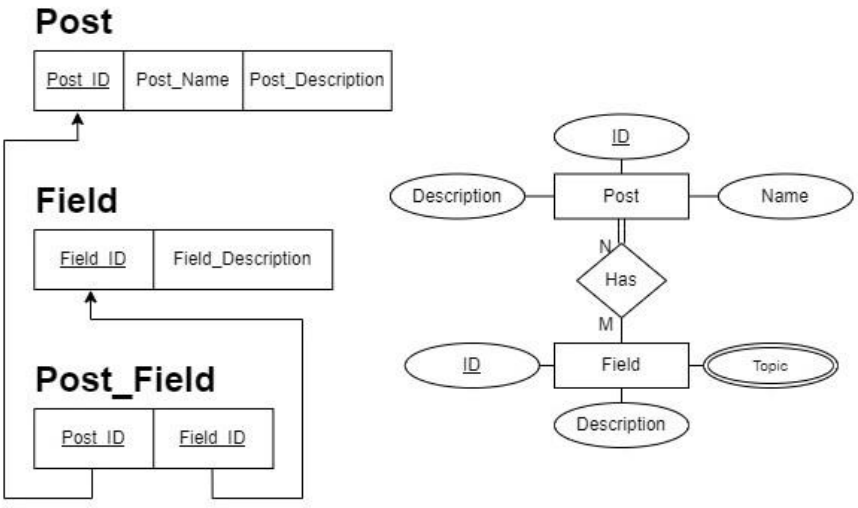


3.5 Mapping of Binary M-N Relationship Types

To map binary M-N relationship types, we used the "cross reference approach", where we created a new relation, add the Primary Keys of the two relationships in the new relation as a Foreign Keys, the Primary Key of this new relation is the combination of the Primary Keys of the two entities.







Session

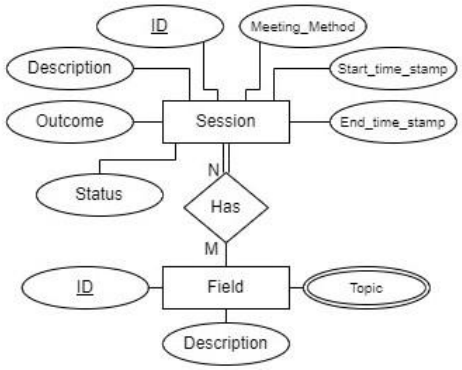
Session_ID	Session_Description	Session_Meeting_Method	Session_Outcome	Start_time_stamp	End_time_stamp	Session_Status

Field

Field_ID	Field_Description

Session_Field

Session_ID	Field_ID



User

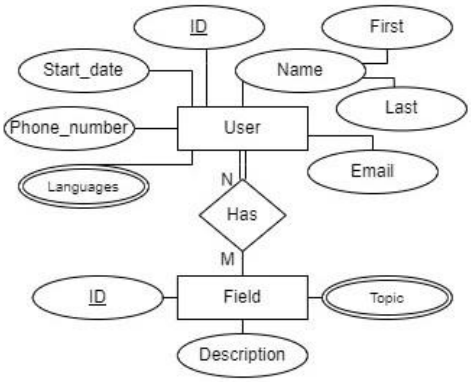
User_ID	User_First_Name	User_Last_Name	Phone_Number	Email	User_Start_Date

Field

Field_ID	Field_Description

User_Field

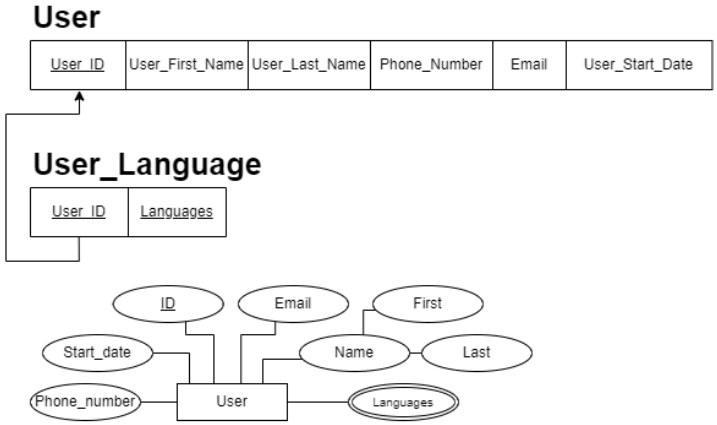
User_ID	Field_ID



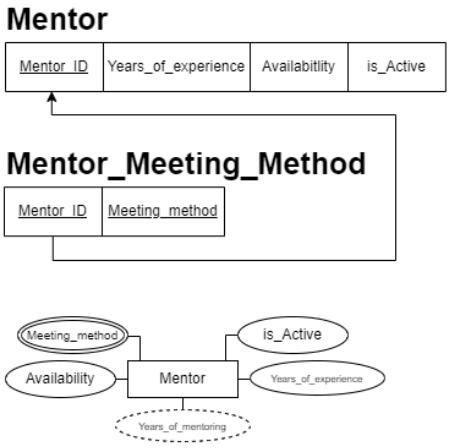
3.6 Mapping of Multivalued Attributes

Multivalued attribute is used when there is more than one value to one attribute. We have three multivalued attributes.

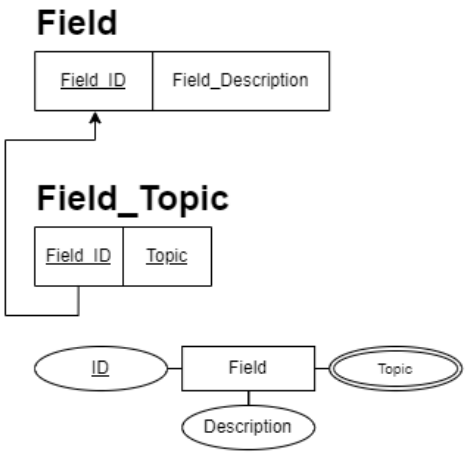
The first one is, [**languages]** which is one of the attributes of [**User**] entity. While [**user_ID**] as foreign key represents the primary key of the User relation. The primary key of the [**User_Language**] relation is combination of {**user_ID, language**}.



The second one is, [**meeting_method]** which is one of the attributes of [**Mentor]** entity. While [**mentor_ID**] as foreign key represents the primary key of the [**Mentor]** relation. The primary key of the [**Mentor_ meeting_method]** relation is combination of {**mentor_ID, meeting_method**}.

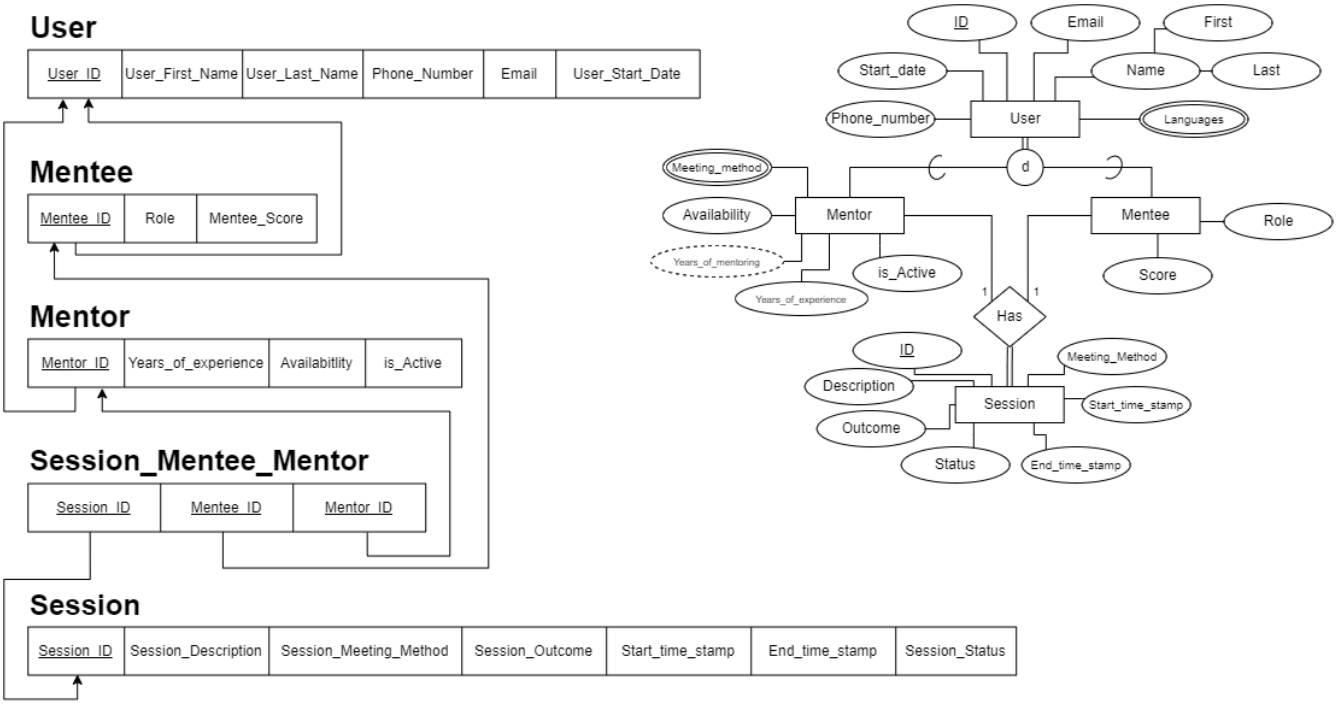


The third one is, [**Topic]** which is one of the attributes of [**Field**] entity. While [**Field _ID**] as foreign key represents the primary key of the [**Field]** relation. The primary key of the [**Field _Topic**] relation is combination of {**Field _ID, topic**}.

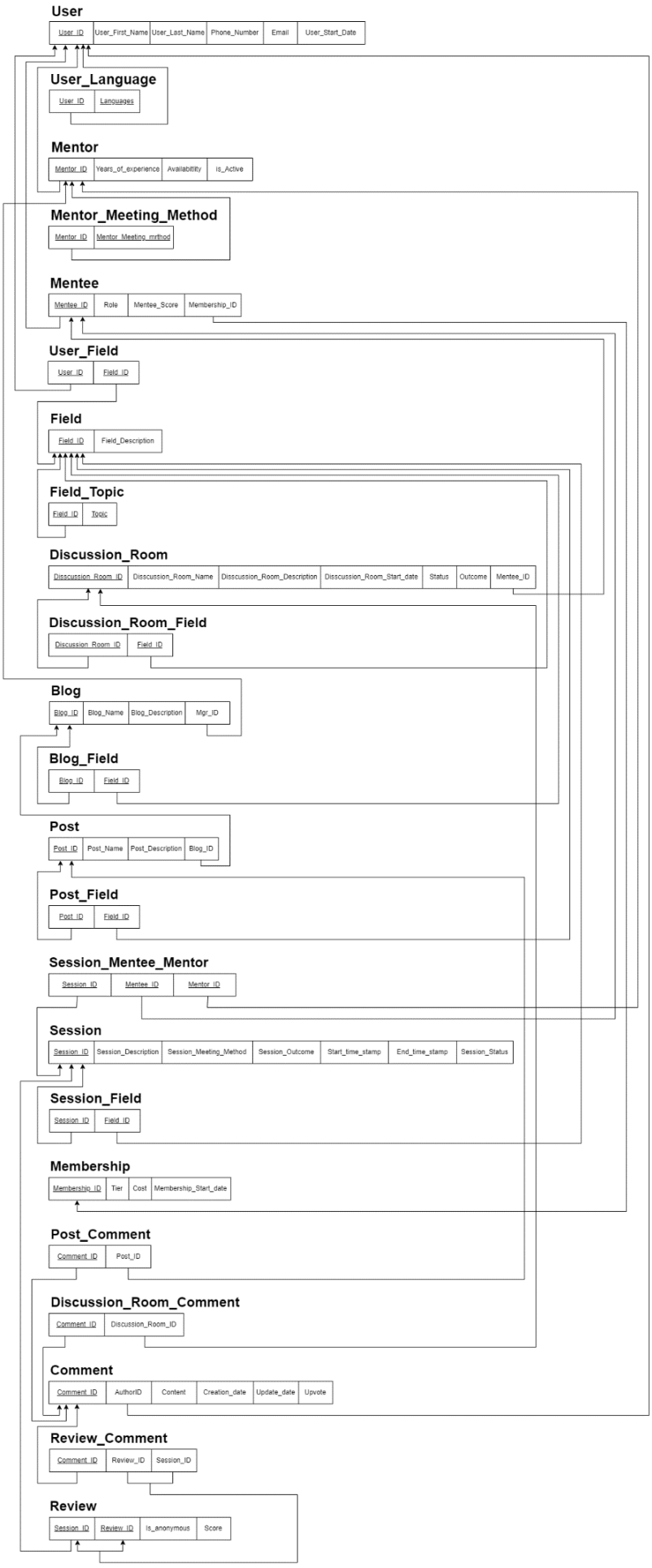


3.7 Mapping of N-ary Relationship Types

The [**Session**] entity must have one [**Mentee**] and one [**Mentor**] at the same time.



3.9 Schema Diagram



4 Normalization

4.1 First Normal Form

To find out whether the relations are in first normal form (1NF), it must not contain composite, multivalued attributes, as well as nested relations. It only includes attributes with atomic values (simple, indivisible) in their domains. In our relational schema, all attributes are atomic, and all attributes depend on the **key**. So, we can conclude that our relations do not violate any guidelines for the first normal form (1NF).

User

User_ID	User_First_Name	User_Last_Name	Phone_Number	Email	User_Start_Date

User_Language

User_ID	Languages

Mentor

Mentor_ID	Years_of_experience	Availabitlity	is_Active

Mentor_Meeting_Method

Mentor_ID	Mentor_Meeting_mrthod

Mentee

Mentee_ID	Role	Mentee_Score	Membership_ID

User_Field

User_ID	Field_ID

Field

Field_ID	Field_Description

Field_Topic

Field_ID	Topic

Blog

Blog_ID	Blog_Name	Blog_Description	Mgr_ID

Blog_Field

Blog_ID	Field_ID

Post

Post_ID	Post_Name	Post_Description	Blog_ID

Post_Field

Post_ID	Field_ID

Discussion_Room

Disscussion_Room_ID	Disscussion_Room_Name	Disscussion_Room_Description	Disscussion_Room_Start_date	Status	Outcome	Mentee_ID

Discussion_Room_Field

Discussion_Room_ID	Field_ID

Session_Mentee_Mentor

Session_ID	Mentee_ID	Mentor_ID

Session

Session_ID	Session_Description	Session_Meeting_Method	Session_Outcome	Start_time_stamp	End_time_stamp	Session_Status

Session_Field

Session_ID	Field_ID

Membership

Membership_ID	Tier	Cost	Membership_Start_date

Post_Comment

Comment_ID	Post_ID

Discussion_Room_Comment

Comment_ID	Discussion_Room_ID

Comment

Comment_ID	AuthorID	Content	Creation_date	Update_date	Upvote

Review_Comment

Comment_ID	Review_ID	Session_ID

Review

Session_ID	Review_ID	Is_anonymous	Score

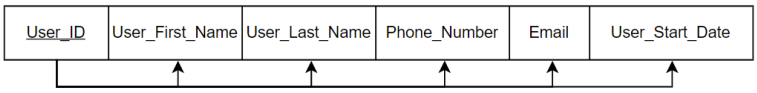
4.2 Second Normal Form

To be in the second normal form (2NF), a relation must be in the first normal form and the relation must not contain any partial dependency.

In the previous step, we verified that the relations are in the first normal form. In this step, we should make sure that every non-primary-key attribute is **fully functionally dependent** on the primary key (No Partial Dependency).

Full FD: means that the removal of the primary key attribute (or an attribute that is part of a PK) results in losing the functional dependency.

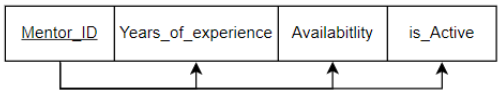
User



{User_ID} -> User_First_Name, User_Last_Name,phone,Number,Email,User_startDate

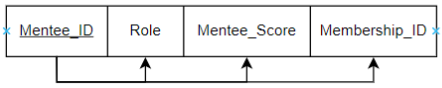
all functionally dependent [FD] on the Primary key, so it is a full FD.

Mentor



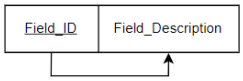
{Mentor_ID} -> Years_of_experience, Availability, is_Active are full FD.

Mentee



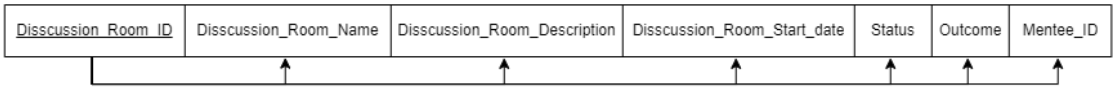
{Mentee_ID} -> Role,Mentee_Score, Membership are full FD.

Field



{Field_ID} -> Field_Description is full FD.

Discussion_Room

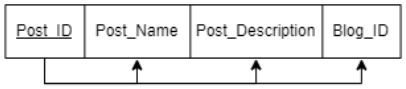


{Disscussion_Room_ID} -> Disscussion_Room_Name, Disscussion_Room_Description, Disscussion_Room_Start_date, Status, Outcome, Mentee_ID are full FD.

Blog

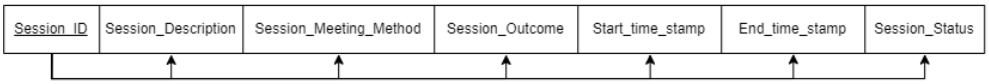
{Blog_ID} -> Blog_Name,Blog_Description, Mgr_ID are full FD.

Post



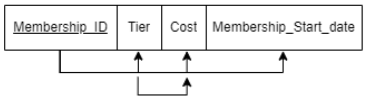
{Post_ID} -> Post_Name,Post_Description, Blog_ID are full FD.

Session



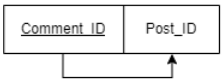
{Session_ID} -> Session_Description, Session _Meeting_Method, Session_Outcome, Start_time_stamp, End_time_stamp, Session_status are full FD.

Membership



{Membership_ID} -> Tier, Cost, Membership_Start_date is full FD.

Post_Comment

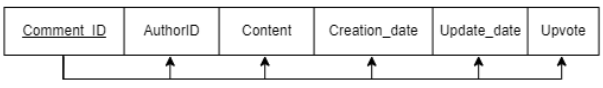


Discussion_Room_Comment

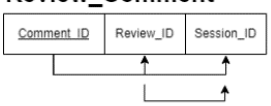
{Comment_ID}->Post_ID is a full FD,

{Comment_ID}->Discussion_Room_ID is a full FD.

Comment



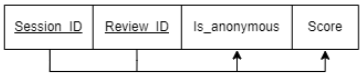
Review_Comment



{Comment_ID}->Author_ID, Content, Creation_date, Update_date, Upvot are a full FD,

{Comment_ID}->Review_ID, session_ID are a full FD.

Review



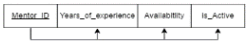
{Session_ID}->Review_ID, Is_anonymous, Score are a full FD.

All attributes depend on the whole key (NO PARTIAL DEPENDENCY), So, all relations are in 2NF.

User

User_Language

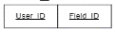
Mentor



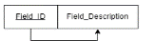
Mentor_Meeting_Method

Mentee

User_Field

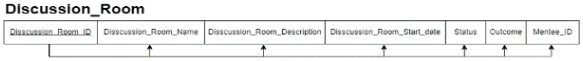


Field



Field_Topic

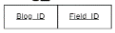
Discussion_Room



Discussion_Room_Field

Blog

Blog_Field



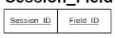
Post

Post_Field

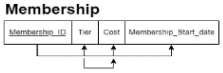
Session_Mentee_Mentor

Session

Session_Field

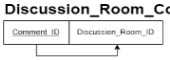


Membership

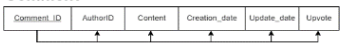


Post_Comment

Discussion_Room_Comment



Comment



Review_Comment

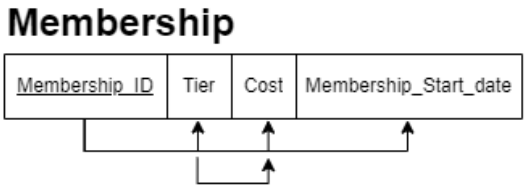
Review

4.3 Third Normal Form

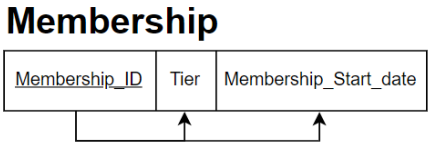
To make our relational schema in the Third normal form (3NF), a relation must be in 2NF and no transitive dependency for non-prime attributes.

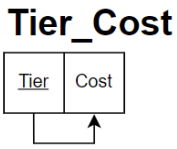
In the previous step, we verified that the relations are in the second normal form. In this step, we should make sure that all attributes must depend on nothing but the key.

In our relations, Membership is not in 3NF since Tier is a non-prime attribute and Cost is a non-prime attribute.

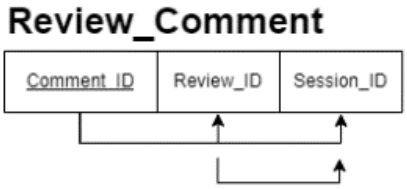


To normalize **Membership** into 3NF:

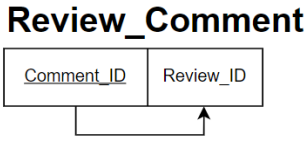


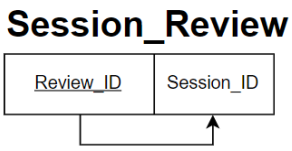


Also, Review_comment is violating the 3NF since Review_ID is a non-prime attribute and Session_ID is a non-prime attribute



To normalize **Review_Comment** into 3NF:

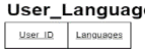




Now no transitive dependency in our relations, so all relation schemas are in 3NF.

User

User_Language



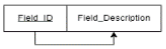
Mentor

Mentor_Meeting_Method

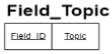
Mentee

User_Field

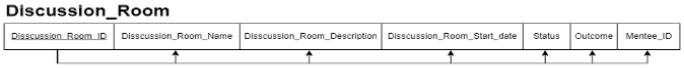
Field



Field_Topic



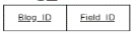
Discussion_Room



Discussion_Room_Field

Blog

Blog_Field

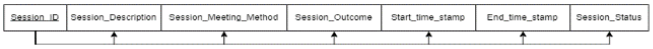


Post

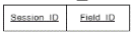
Post_Field

Session_Mentee_Mentor

Session



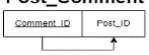
Session_Field



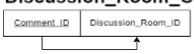
Membership

Tier_Cost

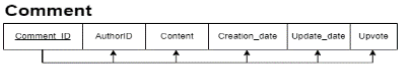
Post_Comment



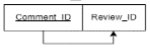
Discussion_Room_Comment



Comment

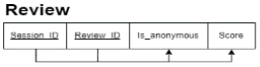


Review_Comment

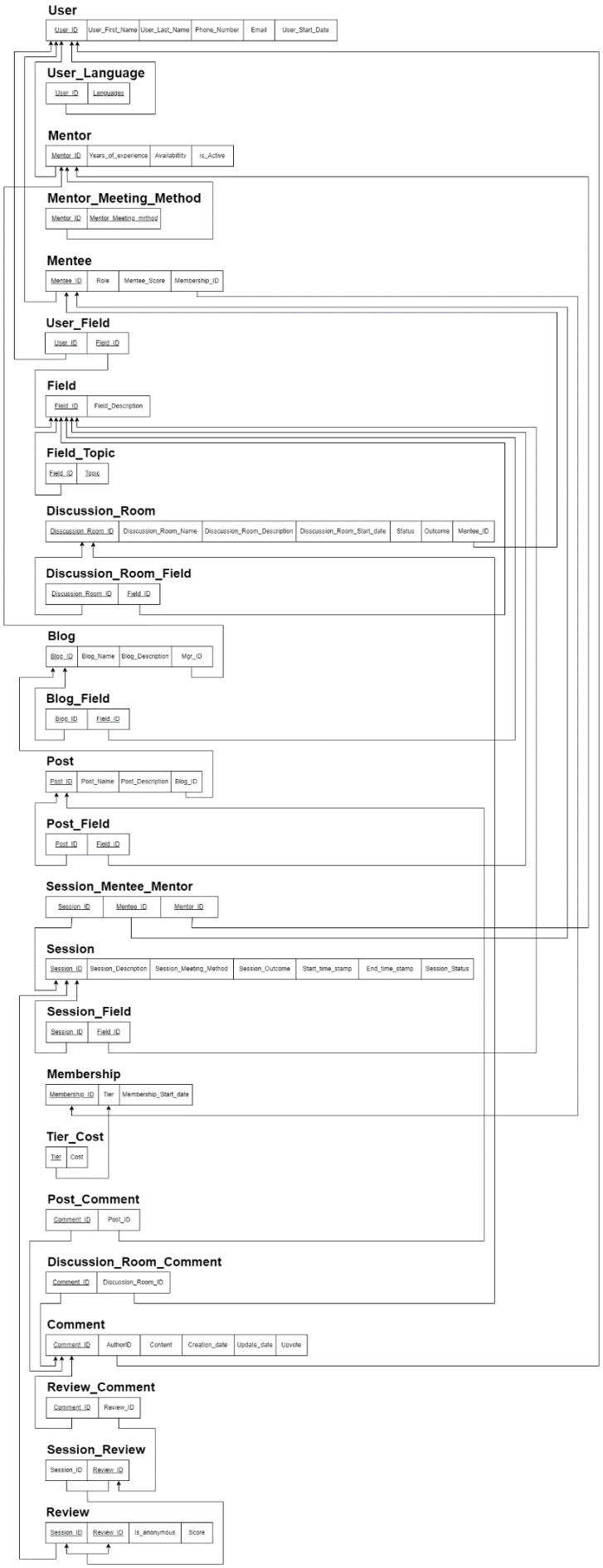


Session_Review

Review



5 Final DB Schema Diagram



PART III: IMPLEMENTATION

6 Table Creation Script

6.1 <User_pf> TABLE

```
create table User_pf (
User_ID number (10) primary key,
User_First_Name varchar2 (50 char) not null,
User_Last_Name varchar2 (50 char) not null,
Phone_Number number (10) not null,
Email varchar2 (64) not null,
User_Start_Date date not null
);
```

```
2  create table User_pf (
3  User_ID number (10) primary key,
4  User_First_Name varchar2 (50 char) not null,
5  User_Last_Name varchar2 (50 char) not null,
6  Phone_Number number (10) not null,
7  Email varchar2 (64) not null,
8  User_Start_Date date not null
9  );
```

6.2 <User_Language> TABLE

```
create table User_Language (
User_ID number (10) not null,
Language varchar2(50),
CONSTRAINT Pk_User_Language Primary Key (User_ID, Language)
);
```

```
13  create table User_Language (
14  User_ID number (10) not null,
15  Language varchar2(50),
16  CONSTRAINT Pk_User_Language Primary Key (User_ID, Language)
17  );
```

```
alter table User_Language
ADD CONSTRAINT FK_User_ID Foreign key (User_ID) references User_pf(User_ID)on
delete cascade;
```

```
165  alter table User_Language
166  ADD CONSTRAINT FK_User_ID Foreign key (User_ID) references User_pf(User_ID)on delete cascade;
```

6.3 <Mentor> TABLE

```
create table Mentor(
Mentor_ID number (10) Primary Key,
Years_of_experience number (2) check (Years_of_experience > 0),
Availability number (1) not NULL,
is_Active number (1) not NULL
);
```

```
20  create table Mentor(
21  Mentor_ID number (10) Primary Key,
22  Years_of_experience number (2) check (Years_of_experience > 0),
23  Availability number (1) not NULL,
24  is_Active number (1) not NULL
25  );
```

```
alter table Mentor
ADD CONSTRAINT FK_Mentor_ID Foreign key (Mentor_ID) references User_pf
(User_ID)on delete cascade;
```

```
169 alter table Mentor
170 ADD CONSTRAINT FK_Mentor_ID Foreign key (Mentor_ID) references User_pf (User_ID)on delete cascade;
```

6.4 <Mentor_Meeting_Method> TABLE

```
create table Mentor_Meeting_Method(
Mentor_ID number (10) ,
Mentor_Meeting_Method varchar2 (100) not null,
CONSTRAINT Pk_Mentor_Meeting_Method Primary Key (Mentor_ID,
Mentor_Meeting_Method)
);
```

```
28 create table Mentor_Meeting_Method(
29 Mentor_ID number (10) ,
30 Mentor_Meeting_Method varchar2 (100) not null,
31 CONSTRAINT Pk_Mentor_Meeting_Method Primary Key (Mentor_ID, Mentor_Meeting_Method)
32 );
```

```
alter table Mentor_Meeting_Method
ADD CONSTRAINT FK_Mentor_ID2 Foreign key (Mentor_ID) references Mentor
(Mentor_ID)on delete cascade;
```

```
173 alter table Mentor_Meeting_Method
174 ADD CONSTRAINT FK_Mentor_ID2 Foreign key (Mentor_ID) references Mentor (Mentor_ID)on delete cascade;
```

6.5 <Mentee> TABLE

```
create table Mentee(
Mentee_ID number (10) Primary Key,
Role varchar2 (20 ) not null,
Mentee_Score number(10) not null,
Membership_ID number (10) not null
);
```

```
34 create table Mentee(
35 Mentee_ID number (10) Primary Key,
36 Role varchar2 (20 ) not null,
37 Mentee_Score number(10) not null,
38 Membership_ID number (10) not null
39 );
```

```
alter table Mentee
ADD CONSTRAINT FK_Mentee_ID Foreign key (Mentee_ID) references User_pf
(User_ID)on delete cascade;
```

```
alter table Mentee
ADD CONSTRAINT FK_Membership_ID Foreign key (Membership_ID) references Membership
(Membership_ID)on delete cascade;
```

```
177 alter table Mentee
178 ADD CONSTRAINT FK_Mentee_ID Foreign key (Mentee_ID) references User_pf (User_ID)on delete cascade;
179 alter table Mentee
180 ADD CONSTRAINT FK_Membership_ID Foreign key (Membership_ID) references Membership (Membership_ID)on delete cascade;
```

6.6 <User_Field > TABLE

```
create table User_Field )
User_ID number(10)Not null,
Field_ID number(10) Not null,
CONSTRAINT PK_User_Field  Primary Key (User_ID, Field_ID));
```

```
1  create table User_Field (
2  User_ID number(10)Not null,
3  Field_ID number(10) Not null,
4  CONSTRAINT PK_User_Field  Primary Key (User_ID, Field_ID));
5
```

```
alter table User_Field
ADD CONSTRAINT FK_User_ID2 Foreign key (User_ID) references User_pf(User_ID)on
delete cascade;
```

```
37 alter table User_Field
38 ADD CONSTRAINT FK_User_ID2 Foreign key (User_ID) references User_pf(User_ID)on delete cascade;
39
```

```
alter table User_Field
ADD CONSTRAINT FK_Field_ID Foreign key (Field_ID) references Field(Field_ID)on
delete cascade;
```

```
40 alter table User_Field
41 ADD CONSTRAINT FK_Field_ID Foreign key (Field_ID) references Field(Field_ID)on delete cascade;
42
```

6.7 <Field > TABLE

```
create table Field (
Field_ID number(10) Primary Key,
Field_Description varchar2(1000)
);
```

```
6  create table Field (
7  Field_ID number(10) Primary Key,
8  Field_Description varchar2(1000)
9  );
10
```

6.8 <Field_Topic> TABLE

```
create table Field_Topic(
Field_ID number(10) Not null ,
Topic varchar2(30) Not null,
CONSTRAINT PK_Field_Topic  Primary Key (Topic, Field_ID));
```

```
11 create table Field_Topic(
12 Field_ID number(10) Not null ,
13 Topic varchar2(30) Not null,
14 CONSTRAINT PK_Field_Topic  Primary Key (Topic, Field_ID));
15
```

```
alter table Field_Topic
ADD CONSTRAINT FK_Field_ID2 Foreign key (Field_ID) references Field(Field_ID)on
delete cascade;
```

```
43 alter table Field_Topic
44 ADD CONSTRAINT FK_Field_ID2 Foreign key (Field_ID) references Field(Field_ID)on delete cascade;
45
```

6.9 <Discussion_Room> TABLE

```
create table Discussion_Room(
Discussion_Room_ID number(10) Primary Key,
Discussion_Room_name varchar2(100) Not null,
Discussion_Room_Description varchar2(1000),
Outcome varchar(20),
Discussion_Room_Start_date  date Not null ,
Status varchar(20),
Mentee_ID number(10) Not null);
```

```
16  create table Discussion_Room(
17   Discussion_Room_ID number(10) Primary Key,
18   Discussion_Room_name varchar2(100) Not null,
19   Discussion_Room_Description varchar2(1000),
20  Outcome varchar(20),
21   Discussion_Room_Start_date  date Not null ,
22  Status varchar(20),
23  Mentee_ID number(10) Not null);
```

```
alter table  Discussion_Room
ADD CONSTRAINT FK_Mentee_ID2 Foreign key ( Mentee_ID) references
Mentee(Mentee_ID)on delete cascade;
```

```
46  alter table  Discussion_Room
47  ADD CONSTRAINT FK_Mentee_ID2 Foreign key ( Mentee_ID) references Mentee(Mentee_ID)on delete cascade;
48
```

610 < Discussion_Room_Field> TABLE

```
create table  Discussion_Room_Field(
Discussion_Room_ID number(10),
Field_ID number(10),
CONSTRAINT PK_Discussion_Room_Field Primary Key (Discussion_Room_ID, Field_ID));
```

```
25  create table  Discussion_Room_Field(
26  Discussion_Room_ID number(10),
27  Field_ID number(10),
28  CONSTRAINT PK_Discussion_Room_Field Primary Key (Discussion_Room_ID, Field_ID));
29
```

```
alter table  Discussion_Room_Field
ADD CONSTRAINT FK_Discussion_Room_ID Foreign key (  Discussion_Room_ID)
references Discussion_Room( Discussion_Room_ID)on delete cascade;
```

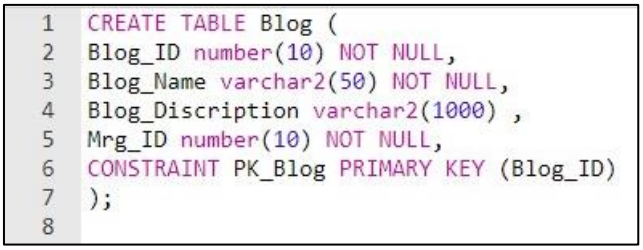
```
49  alter table  Discussion_Room_Field
50  ADD CONSTRAINT FK_Discussion_Room_ID Foreign key (  Discussion_Room_ID) references Discussion_Room( Discussion_Room_ID)on delete cascade;
51
```

```
alter table  Discussion_Room_Field
ADD CONSTRAINT FK_Field_ID3 Foreign key (  Field_ID) references Field(Field_ID)on
delete cascade;
```

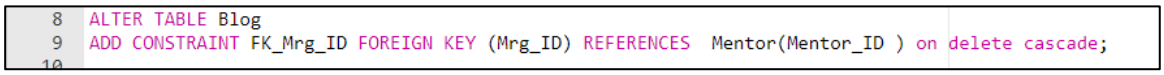
```
52  alter table  Discussion_Room_Field
53  ADD CONSTRAINT FK_Field_ID3 Foreign key (  Field_ID) references Field(Field_ID)on delete cascade;
54
```

6.11 <Blog> TABLE

```
CREATE TABLE Blog (
Blog_ID number(10) NOT NULL,
Blog_Name varchar2(50) NOT NULL,
Blog_Discription varchar2(1000) ,
Mrg_ID number(10) NOT NULL,
CONSTRAINT PK_Blog PRIMARY KEY (Blog_ID)
);
```

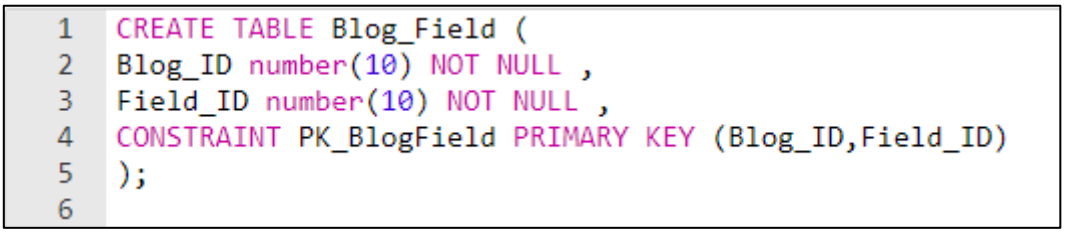



```
ALTER TABLE Blog
ADD CONSTRAINT FK_Mrg_ID FOREIGN KEY (Mrg_ID) REFERENCES  Mentor(Mentor_ID ) on
delete cascade;
```

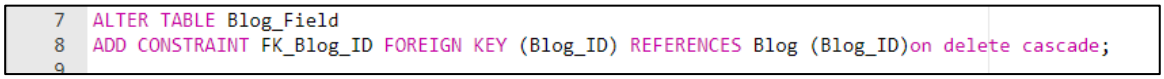


6.12 <Blog_Field> TABLE

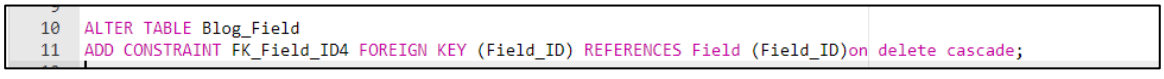
```
CREATE TABLE Blog_Field (
Blog_ID number(10) NOT NULL ,
Field_ID number(10) NOT NULL ,
CONSTRAINT PK_BlogField PRIMARY KEY (Blog_ID,Field_ID)
);
```




```
ALTER TABLE Blog_Field
ADD CONSTRAINT FK_Blog_ID FOREIGN KEY (Blog_ID) REFERENCES Blog (Blog_ID) on
delete cascade;
```

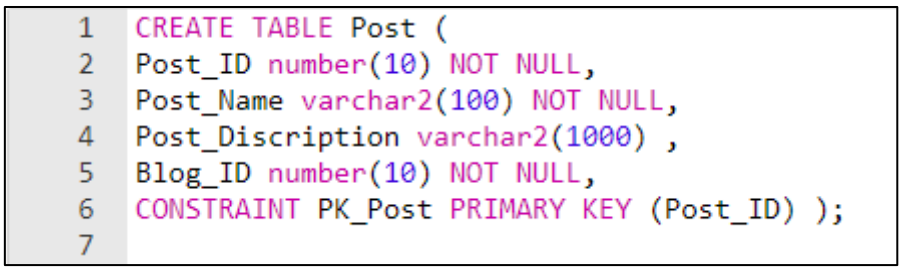



```
ALTER TABLE Blog_Field
ADD CONSTRAINT FK_Field_ID4 FOREIGN KEY (Field_ID) REFERENCES Field (Field_ID)on
delete cascade;
```

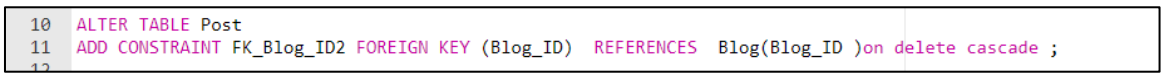


6.13 <Post> TABLE

```
CREATE TABLE Post (
Post_ID number(10) NOT NULL,
Post_Name varchar2(100) NOT NULL,
Post_Discription varchar2(1000) ,
Blog_ID number(10) NOT NULL,
CONSTRAINT PK_Post PRIMARY KEY (Post_ID) );
```

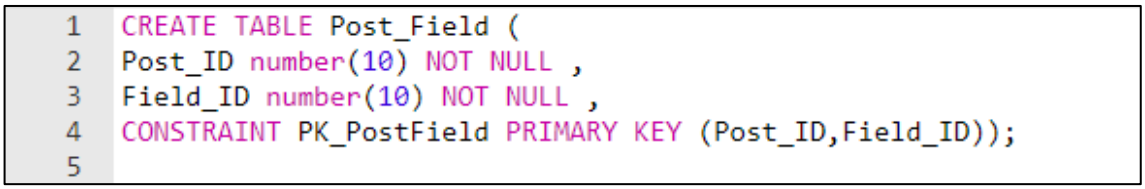



```
ALTER TABLE Post
ADD CONSTRAINT FK_Blog_ID2 FOREIGN KEY (Blog_ID)  REFERENCES  Blog(Blog_ID )on
delete cascade ;
```

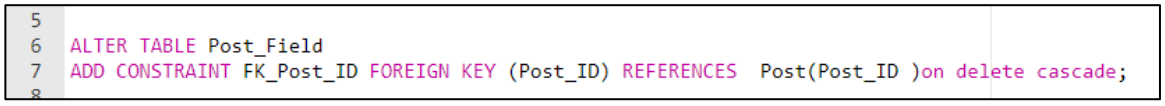


6.14 <Post_Field > TABLE

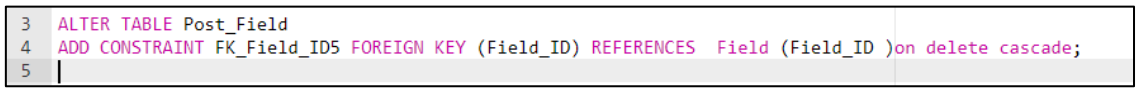
```
CREATE TABLE Post_Field (
Post_ID number(10) NOT NULL ,
Field_ID number(10) NOT NULL ,
CONSTRAINT PK_PostField PRIMARY KEY (Post_ID,Field_ID));
```




```
ALTER TABLE Post_Field
ADD CONSTRAINT FK_Post_ID FOREIGN KEY (Post_ID) REFERENCES  Post(Post_ID )on
delete cascade;
```

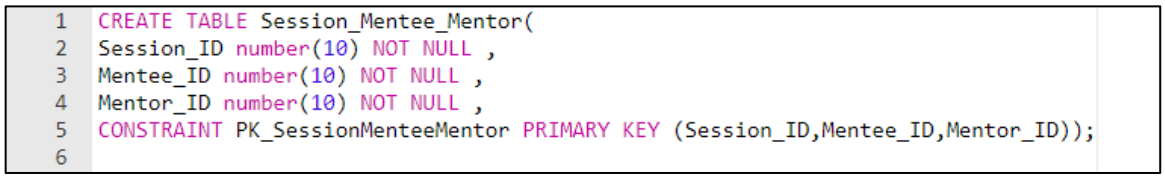



```
ALTER TABLE Post_Field
ADD CONSTRAINT FK_Field_ID5 FOREIGN KEY (Field_ID) REFERENCES  Field (Field_ID
)on delete cascade;
```

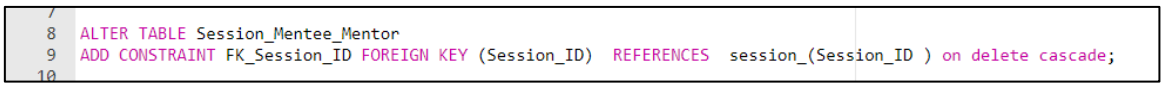


6.15 <Session_Mentee_Mentor> TABLE

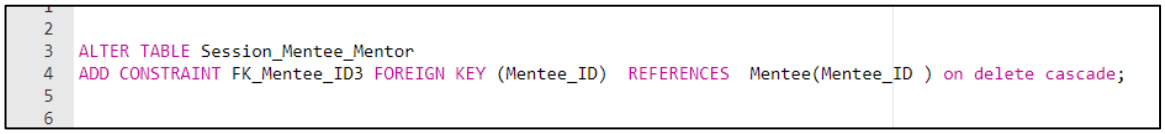
```
CREATE TABLE Session_Mentee_Mentor(
Session_ID number(10) NOT NULL ,
Mentee_ID number(10) NOT NULL ,
Mentor_ID number(10) NOT NULL ,
CONSTRAINT PK_SessionMenteeMentor PRIMARY KEY (Session_ID,Mentee_ID,Mentor_ID));
```



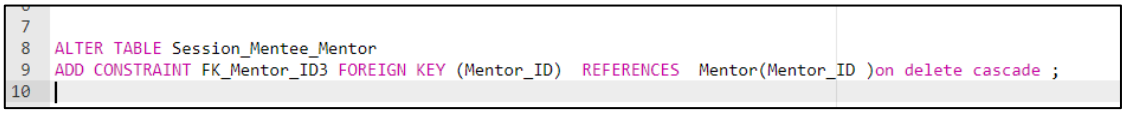

```
ALTER TABLE Session_Mentee_Mentor
ADD CONSTRAINT FK_Session_ID FOREIGN KEY (Session_ID)  REFERENCES
session_(Session_ID ) on delete cascade;
```




```
ALTER TABLE Session_Mentee_Mentor
ADD CONSTRAINT FK_Mentee_ID3 FOREIGN KEY (Mentee_ID)  REFERENCES
Mentee(Mentee_ID ) on delete cascade;
```

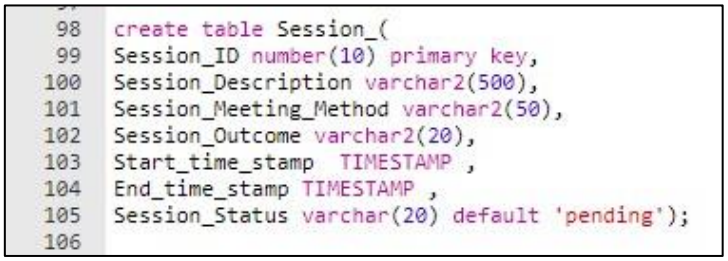



```
ALTER TABLE Session_Mentee_Mentor
ADD CONSTRAINT FK_Mentor_ID3 FOREIGN KEY (Mentor_ID)  REFERENCES
Mentor(Mentor_ID )on delete cascade ;
```



6.16 <Session_> TABLE

```
create table Session_(
Session_ID number(10) primary key,
Session_Description varchar2(500),
Session_Meeting_Method varchar2(50),
Session_Outcome varchar2(20),
Start_time_stamp  TIMESTAMP ,
End_time_stamp TIMESTAMP ,
Session_Status varchar(20) default 'pending');
```



6.17 <Session_Feild> TABLE

```
create table Session_Feild(
Session_ID number(10),
Feild_ID number(10),
CONSTRAINT PK_SessionFeild Primary Key(Session_ID, Feild_ID
));
```

```
107  create table Session_Feild(
108  Session_ID number(10),
109  Feild_ID number(10),
110  CONSTRAINT PK_SessionFeild Primary Key(Session_ID, Feild_ID));
```

```
alter table session_feild
ADD CONSTRAINT FK_Session_ID2 Foreign key (Session_ID) references Session_
(Session_ID)on delete cascade;
alter table session_feild
ADD CONSTRAINT FK_Field_ID6 Foreign key (Feild_ID) references Field (Field_ID)on
delete cascade;
```

```
206  alter table session_feild
207  ADD CONSTRAINT FK_Session_ID2 Foreign key (Session_ID) references Session_ (Session_ID)on delete cascade;
208  alter table session_feild
209  ADD CONSTRAINT FK_Field_ID6 Foreign key (Feild_ID) references Field (Field_ID)on delete cascade;
210
```

6.18 <Membership> TABLE

```
create table Membership(
Membership_ID number(10) primary key,
tier varchar(20),
Membership_Start_date date);
```

```
111
112  create table Membership(
113  Membership_ID number(10) primary key,
114  tier varchar(20),
115  Membership_Start_date date);
116
```

```
alter table membership
ADD CONSTRAINT FK_Tier_Cost Foreign key(tier) references tier_cost(tier)on delete
cascade;
```

```
218  alter table membership
219  ADD CONSTRAINT FK_Tier_Cost Foreign key(tier) references tier_cost(tier)on delete cascade;
220
```

6.19 <Tier_Cost> TABLE

```
create table Tier_Cost(
tier varchar(20) primary key,
Cost number(2));
```

```
119  create table Tier_Cost(
120  tier varchar(20) primary key,
121  Cost number(2));
122
```

6.20 <Post_Comment> TABLE

```
create table Post_Comment(
Comment_ID number(10) primary key,
Post_ID number(10));
```

```
123  create table Post_Comment(
124  Comment_ID number(10) primary key,
125  Post_ID number(10));
126
```

```
alter table Post_Comment
ADD CONSTRAINT FK_Comment_ID Foreign key(Comment_ID) references Comment_
(Comment_ID)on delete cascade;
alter table Post_Comment
ADD CONSTRAINT FK_Post_ID2 Foreign key(Post_ID) references Post(Post_ID)on delete
cascade;
```

```
212  alter table Post_Comment
213  ADD CONSTRAINT FK_Comment_ID Foreign key(Comment_ID) references Comment_ (Comment_ID)on delete cascade;
214  alter table Post_Comment
215  ADD CONSTRAINT FK_Post_ID2 Foreign key(Post_ID) references Post(Post_ID)on delete cascade;
216
```

6.21 < Discussion_Room_Comment> TABLE

```
create table Discussion_Room_Comment (
Comment_ID number(10) Primary Key,
Discussion_Room_ID number(10) not NULL
);
```

```
130  create table Discussion_Room_Comment (
131  Comment_ID number(10) Primary Key,
132  Discussion_Room_ID number(10) not NULL
133  );
```

```
alter table Discussion_Room_Comment
ADD CONSTRAINT FK_Comment_ID2 Foreign key (Comment_ID) references Comment_
(Comment_ID)on delete cascade;
```

```
222  alter table Discussion_Room_Comment
223  ADD CONSTRAINT FK_Comment_ID2 Foreign key (Comment_ID)
224  references Comment_ (Comment_ID) on delete cascade;
```

```
alter table Discussion_Room_Comment
ADD CONSTRAINT FK_Discussion_Room_ID2 Foreign key (Discussion_Room_ID) references
Discussion_Room (Discussion_Room_ID)on delete cascade;
```

```
226  alter table Discussion_Room_Comment
227  ADD CONSTRAINT FK_Discussion_Room_ID2 Foreign key (Discussion_Room_ID)
228  references Discussion_Room (Discussion_Room_ID) on delete cascade;
```

6.22 <Comment_> TABLE

```
create table Comment_ (
Comment_ID number(10) Primary Key,
AuthorID number(10) not NULL,
Content varchar2(500 char) not NULL,
Creation_date date not NULL,
Update_date date not NULL,
Upvote number(3)
);
```

```
135  create table Comment_ (
136  Comment_ID number(10) Primary Key,
137  AuthorID number(10) not NULL,
138  Content varchar2(500 char) not NULL,
139  Creation_date date not NULL,
140  Update_date date not NULL,
141  Upvote number(3)
142  );
```

```
alter table Comment_
ADD CONSTRAINT FK_AuthorID Foreign key (AuthorID) references User_pf (User_ID)on
delete cascade;
```

```
230  alter table Comment_
231  ADD CONSTRAINT FK_AuthorID Foreign key (AuthorID)
232  references User_pf (User_ID) on delete cascade;
```

6.22 <Review_Comment> TABLE

```
create table Review_Comment (
Comment_ID number(10) Primary Key,
Review_ID number(10) not NULL
);
```

```
144  create table Review_Comment (
145  Comment_ID number(10) Primary Key,
146  Review_ID number(10) not NULL
147  );
```

```
alter table Review_Comment
ADD CONSTRAINT FK_Comment_ID3 Foreign key (Comment_ID) references Comment_
(Comment_ID)on delete cascade;
```

```
234  alter table Review_Comment
235  ADD CONSTRAINT FK_Comment_ID3 Foreign key (Comment_ID)
236  references Comment_ (Comment_ID) on delete cascade;
```

```
alter table Review_Comment
ADD CONSTRAINT FK_Review_ID2 Foreign key (Review_ID) references Session_Review
(Review_ID)on delete cascade;
```

```
238  alter table Review_Comment
239  ADD CONSTRAINT FK_Review_ID2 Foreign key (Review_ID)
240  references Session_Review (Review_ID) on delete cascade;
```

6.23 <Session_Review> TABLE

```
create table Session_Review (
Session_ID number(10),
Review_ID number(10) Primary Key
);
```

```
149  create table Session_Review (
150  Session_ID number(10),
151  Review_ID number(10) Primary Key
152  );
```

```
alter table Session_Review
ADD CONSTRAINT FK_Session_ID_Review_ID  Foreign key (Session_ID, Review_ID)
references Review (Session_ID, Review_ID)on delete cascade;
```

```
242  alter table Session_Review
243  ADD CONSTRAINT FK_Session_ID_Review_ID  Foreign key (Session_ID, Review_ID)
244  references Review (Session_ID, Review_ID) on delete cascade;
```

6.24 <Review> TABLE

```
create table Review (
Session_ID number(10),
Review_ID number(10),
Is_anonymous number(1) default 0 not NULL,
Score number(1) not NULL check(Score between 1 AND 5),
CONSTRAINT PK_Review Primary Key (Session_ID, Review_ID)
);
```

```
154  create table Review (
155  Session_ID number(10),
156  Review_ID number(10),
157  Is_anonymous number(1) default 0 not NULL,
158  Score number(1) not NULL check(Score between 1 AND 5),
159  CONSTRAINT PK_Review Primary Key (Session_ID, Review_ID)
160  );
```

```
alter table Review
ADD CONSTRAINT FK_Session_ID3 Foreign key (Session_ID) references Session_
(Session_ID) on delete cascade;
```

```
246   alter table Review
247  ADD CONSTRAINT FK_Session_ID3 Foreign key (Session_ID)
248  references Session_ (Session_ID) on delete cascade;
```

7 Constraints Script

This schedule doesn't contain all the constraints but only the important constraints that we included in the script, and we included an example for each important constraint.

Business Rule	SQL Script	Table
Each User has unique ID	User_ID number (10) **primary key**	User_pf
The default value for status is pending	Session_Status varchar (20) **default 'pending'**	Session_
The default value for is_anonymous is number 0 for false	Is_anonymous number (1) **default 0 not NULL**	Review
Years of experience must be positive number	Years_of_experience number (2) **check (Years_of_experience>0)**	Mentor
Score number must be between 1 and 5	Score number (1) not NULL **check (Score between 1 AND 5)**	Review
A blog cannot be created without a manger (mentor). If a mentor is deleted, the foreign key in all referencing relations is deleted as well	Mrg_ID number (10) **NOT NULL,** ALTER TABLE Blog ADD CONSTRAINT FK_Mrg_ID **FOREIGN KEY (Mrg_ID)** **REFERENCES Mentor (Mentor_ID)** **on delete cascade;**	Blog

8 Queries and Transactions

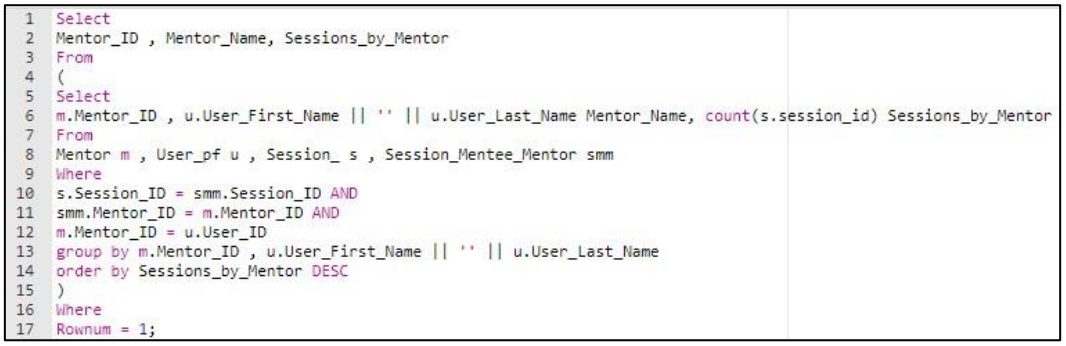
8.1 *<Top mentors by number of sessions>*

Query in Natural Language (English)

This query will retrieve the top mentors by the number of sessions with others. Our database will be used in this context by the users to know which mentor has a lot of experience doing sessions. The mentor with a higher number of sessions is more expert than the mentor with little to no sessions.

SQL Script

```
Select
Mentor_ID , Mentor_Name, Sessions_by_Mentor
From
(
Select
m.Mentor_ID , u.User_First_Name || '' || u.User_Last_Name Mentor_Name,
count(s.session_id) Sessions_by_Mentor
From
Mentor m , User_pf u , Session_ s , Session_Mentee_Mentor smm
Where
s.Session_ID = smm.Session_ID AND
smm.Mentor_ID = m.Mentor_ID AND
m.Mentor_ID = u.User_ID
group by m.Mentor_ID , u.User_First_Name || '' || u.User_Last_Name
order by Sessions_by_Mentor DESC
)
Where
Rownum = 1;
```



Caption of the First Five Rows of the Output

MENTOR_ID	MENTOR_NAME	SESSIONS_BY_MENTOR
1000000005	JuryMola	3

8.2 *<All comments>*

Query in Natural Language (English)

This query will retrieve all the comment that are in the database. The user will use it to get an idea of what the usual comments are, and if they are positive or negative comments.

SQL Script

```
Select
'Post' AS Type,
P.Post_Name AS Title, C.AuthorID || ' ' || U.User_First_Name || ' ' ||
U.User_Last_Name AS Comment_Author,
C.Comment_ID, C.Content AS Comment_, C.Creation_Date AS Publish_Date
From
Post P, Post_Comment PC, Comment_ C, User_pf U
Where
P.Post_ID = PC.Post_ID AND
PC.Comment_ID = C.Comment_ID AND
C.AuthorID = U.User_ID

UNION

Select
'Discussion Room' AS Type,
D.Discussion_Room_Name AS Title, C.AuthorID || ' ' || U.User_First_Name || ' ' ||
U.User_Last_Name AS Comment_Author,
C.Comment_ID, C.Content AS Comment_, C.Creation_Date AS Publish_Date
From
Discussion_Room D, Discussion_Room_Comment DC, Comment_ C, User_pf U
Where
D.Discussion_Room_ID = DC.Discussion_Room_ID AND
DC.Comment_ID = C.Comment_ID AND
C.AuthorID = U.User_ID

UNION

Select
'Review' AS Type,
'--' AS Title,
DECODE(R.Is_Anonymous, 0 , 'Anonymous' ,
C.AuthorID || ' ' || U.User_First_Name || ' ' || U.User_Last_Name) AS
Comment_Author,
C.Comment_ID, C.Content AS Comment_, C.Creation_Date AS Publish_Date
From
Review R, Review_Comment RC, Comment_ C, User_pf U
Where
R.Review_ID = RC.Review_ID AND
RC.Comment_ID = C.Comment_ID AND
C.AuthorID = U.User_ID;
```

```sql
Select
'Post' AS Type,
P.Post_Name AS Title, C.AuthorID || ' ' || U.User_First_Name || ' ' || U.User_Last_Name AS Comment_Author,
C.Comment_ID, C.Content AS Comment_, C.Creation_Date AS Publish_Date
From
Post P, Post_Comment PC, Comment_ C, User_pf U
Where
P.Post_ID = PC.Post_ID AND
PC.Comment_ID = C.Comment_ID AND
C.AuthorID = U.User_ID

UNION

Select
'Discussion Room' AS Type,
D.Discussion_Room_Name AS Title, C.AuthorID || ' ' || U.User_First_Name || ' ' || U.User_Last_Name AS Comment_Author,
C.Comment_ID, C.Content AS Comment_, C.Creation_Date AS Publish_Date
From
Discussion_Room D, Discussion_Room_Comment DC, Comment_ C, User_pf U
Where
D.Discussion_Room_ID = DC.Discussion_Room_ID AND
DC.Comment_ID = C.Comment_ID AND
C.AuthorID = U.User_ID

UNION

Select
'Review' AS Type,
'Review Of Session: ' || R.Session_ID AS Title,
DECODE(R.Is_Anonymous, 0 , 'Anonymous' ,
C.AuthorID || ' ' || U.User_First_Name || ' ' || U.User_Last_Name) AS Comment_Author,
C.Comment_ID, C.Content AS Comment_, C.Creation_Date AS Publish_Date
From
Review R, Review_Comment RC, Comment_ C, User_pf U
Where
R.Review_ID = RC.Review_ID AND
RC.Comment_ID = C.Comment_ID AND
C.AuthorID = U.User_ID
ORDER BY Publish_Date;
```

Caption of the First Five Rows of the Output

TYPE	TITLE	COMMENT_AUTHOR	COMMENT_ID	COMMENT_	PUBLISH_DATE
Post	Packet Switching Methods on Cisco Networks	1000000000 Sara Al Shareef	8000000017	fairly good	29-JAN-19
Discussion Room	What is a good back end to use with AngularJS	1000000009 Rawan Alyami	8000000001	This was a great explanation	19-APR-19
Post	The power of making game	1000000006 Adam koja	8000000013	Thanks	17-AUG-19
Review	Review Of Session: 6000000001	1000000015 Ahlam Al Magrbi	8000000021	I still need more help	29-JAN-20
Discussion Room	How to open documents folder in iOS swift	1000000011 Maan Al dosory	8000000000	My issue still not fixed	02-FEB-20

8.3 <*Get all reviews for a mentor* >

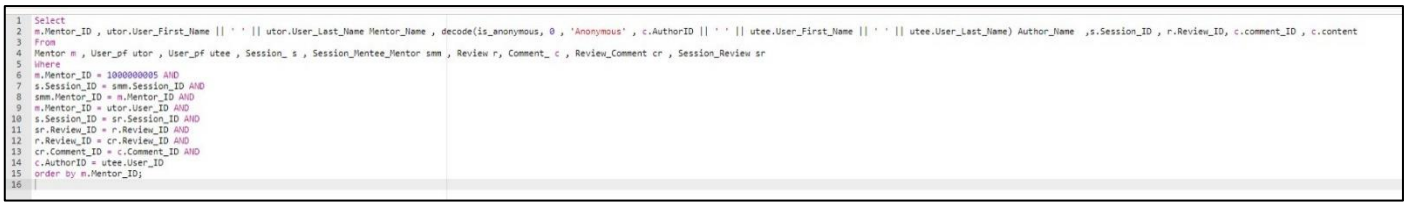
Query in Natural Language (English)

This query will retrieve all the reviews for a specific mentor. The users will use our database in this context to check other people's opinions on the mentor before getting into the session with them.

SQL Script

Get all review for the mentor who has the ID: 1000000005

```
Select
m.Mentor_ID , utor.User_First_Name || ' ' || utor.User_Last_Name Mentor_Name ,
decode(is_anonymous, 0 , 'Anonymous' , c.AuthorID || ' ' || utee.User_First_Name
|| ' ' || utee.User_Last_Name) Author_Name  ,s.Session_ID , r.Review_ID,
c.comment_ID , c.content
From
Mentor m , User_pf utor , User_pf utee , Session_ s , Session_Mentee_Mentor smm ,
Review r, Comment_ c , Review_Comment cr , Session_Review sr
Where
m.Mentor_ID = 1000000005 AND
s.Session_ID = smm.Session_ID AND
smm.Mentor_ID = m.Mentor_ID AND
m.Mentor_ID = utor.User_ID AND
s.Session_ID = sr.Session_ID AND
sr.Review_ID = r.Review_ID AND
r.Review_ID = cr.Review_ID AND
cr.Comment_ID = c.Comment_ID AND
c.AuthorID = utee.User_ID
order by m.Mentor_ID;
```



Caption of the First Five Rows of the Output

MENTOR_ID	MENTOR_NAME	AUTHOR_NAME	SESSION_ID	REVIEW_ID	COMMENT_ID	CONTENT
1000000005	Jury Mola	1000000014 yousof Al harbi	6000000004	9000000004	8000000024	My problem is still there
1000000005	Jury Mola	1000000008 Reem Sobhy	6000000005	9000000005	8000000025	This was a successful solution
1000000005	Jury Mola	1000000019 Ward akmal	6000000006	9000000006	8000000026	Nothing worked

Download CSV

3 rows selected.

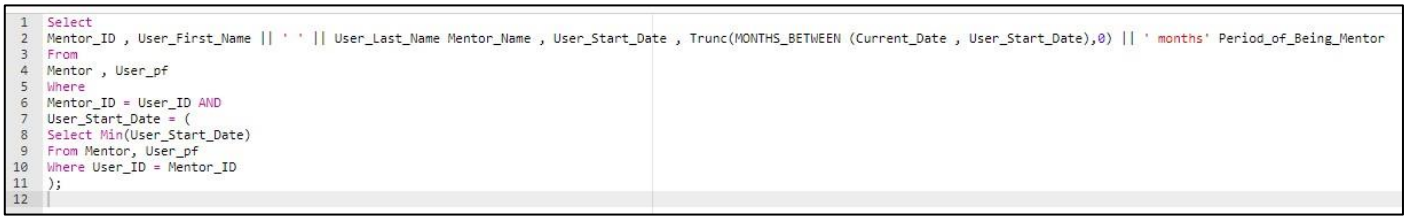
8.4 <*The mentor who has the highest period in being a mentor* >

Query in Natural Language (English)

This query will retrieve the mentor with the highest period in being a mentor. Some users prefer a mentor who isn't new to being a mentor because they want someone used to the environment and the mentoring. The users will use our database in this context to get that mentor.

SQL Script

```
Select
Mentor_ID , User_First_Name || ' ' || User_Last_Name Mentor_Name ,
User_Start_Date , Trunc(MONTHS_BETWEEN (Current_Date , User_Start_Date),0) || '
months' Period_of_Being_Mentor
From
Mentor , User_pf
Where
Mentor_ID = User_ID AND
User_Start_Date = (
Select Min(User_Start_Date)
From Mentor, User_pf
Where User_ID = Mentor_ID
);
```



Caption of the First Five Rows of the Output

MENTOR_ID	MENTOR_NAME	USER_START_DATE	PERIOD_OF_BEING_MENTOR
1000000005	Jury Mola	15-OCT-19	30 months

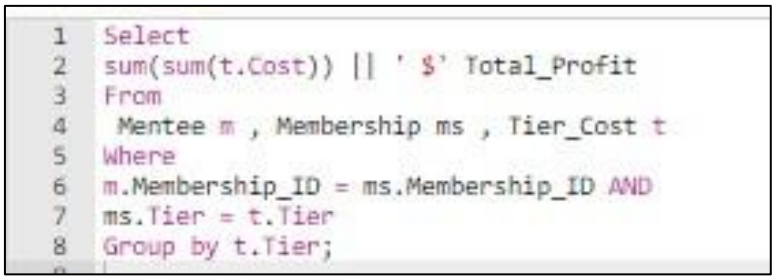
8.5 <*Total profit*>

Query in Natural Language (English)

The query will retrieve the total profit of all memberships. The owners of the website or the database will use this to get the total profit of the website, and how much are they gaining from it.

SQL Script

```
Select
sum(sum(t.Cost)) || ' $' Total_Profit
From
 Mentee m , Membership ms , Tier_Cost t
Where
m.Membership_ID = ms.Membership_ID AND
ms.Tier = t.Tier
Group by t.Tier;
```



Caption of the First Five Rows of the Output

TOTAL_PROFIT
65 $

8.6 Update Example

Update in Natural Language (English)

Update a score for a certain mentee.

SQL Script

```
update mentee Mentee_Score
set Mentee_Score = 200
where Mentee_Id = (select User_id
from user_pf
where User_First_Name= 'Shams' AND User_Last_Name= 'Al bargi' AND Phone_Number=
0576257795
);
```

```
update mentee Mentee_Score
set Mentee_Score = 200
where Mentee_Id = ( select User_id
from user_pf
where User_First_Name= 'Shams' AND User_Last_Name= 'Al bargi' AND Phone_Number= 0576257795
);
```

Caption of the Output

- Before updating:

```
select mentee.Mentee_ID,User_First_Name,User_Last_Name,Phone_Number,Mentee_Score
from mentee,user_pf
where user_pf.user_id =  mentee.Mentee_ID;
```

- The result:

MENTEE_ID	USER_FIRST_NAME	USER_LAST_NAME	PHONE_NUMBER	MENTEE_SCORE
1000000010	layan	Shukor	558565389	80
1000000011	Maan	Al dosory	562873694	65
1000000012	Hadeel	Al ghatani	508675355	90
1000000013	rami	Al bany	509674686	100
1000000014	yousof	Al harbi	508743548	50
1000000015	Ahlam	Al Magrbi	568652977	30
1000000016	Salem	Al Salem	503586846	80
1000000017	rashid	Al thibiti	553896946	80
1000000018	Shams	Al bargi	576257795	80
1000000019	Ward	akmal	505896986	50

Download CSV

10 rows selected.

- After updating:

```
363  select mentee.Mentee_ID,User_First_Name,User_Last_Name,Phone_Number,Mentee_Score
364  from mentee,user_pf
365  where user_pf.user_id =  mentee.Mentee_ID;
366
```

- The result:

MENTEE_ID	USER_FIRST_NAME	USER_LAST_NAME	PHONE_NUMBER	MENTEE_SCORE
1000000010	layan	Shukor	558565389	80
1000000011	Maan	Al dosory	562873694	65
1000000012	Hadeel	Al ghatani	508675355	90
1000000013	rami	Al bany	509674686	100
1000000014	yousof	Al harbi	508743548	50
1000000015	Ahlam	Al Magrbi	568652977	30
1000000016	Salem	Al Salem	503586846	80
1000000017	rashid	Al thibiti	553896946	80
1000000018	Shams	Al bargi	576257795	200
1000000019	Ward	akmal	505896986	50

Download CSV

10 rows selected.

8.7 Delete Example

Delete in Natural Language (English)

Delete sessions that happened before 2022.

SQL Script

```
delete from session_
where END_TIME_STAMP < to_timestamp_tz ('01-JAN-2022 12.00.00 AM');
```

```
550  delete from session_
551  where END_TIME_STAMP < to_timestamp_tz ('01-JAN-2022 12.00.00 AM');
```

Caption of the Output

- Before deleting:

SESSION_ID	SESSION_DESCRIPTION	SESSION_MEETING_METHOD	SESSION_OUTCOME	START_TIME_STAMP	END_TIME_STAMP	SESSION_STATUS
6000000000	-	Zoom	Fully resolved	04-APR-20 08.00.00.000000 PM	04-APR-20 08.40.00.000000 PM	Completed
6000000001	-	Zoom	partially resolved	31-DEC-19 10.30.00.000000 AM	31-DEC-20 03.30.00.000000 PM	Completed
6000000002	How to implement/import graphic design in iOS apps	offline	Not resolved	07-JUL-22 03.04.02.000000 AM	07-JUL-22 05.04.02.000000 PM	Confirmed
6000000003	Video game animation format	google meets	Fully resolved	02-FEB-22 08.29.30.000000 AM	02-FEB-22 09.29.30.000000 AM	Completed
6000000004	-	Zoom	partially resolved	23-AUG-22 09.00.09.000000 PM	23-AUG-22 09.40.09.000000 PM	Confirmed
6000000005	Running node.js in iOS/Android	offline	Fully resolved	23-FEB-21 08.00.08.000000 PM	23-FEB-21 08.40.08.000000 PM	Completed
6000000006	-	google meets	Not resolved	19-SEP-22 01.00.00.000000 PM	19-SEP-22 01.40.00.000000 PM	Confirmed
6000000007	-	offline	partially resolved	23-OCT-22 09.30.45.000000 PM	23-OCT-22 10.30.45.000000 PM	pending
6000000008	-	Zoom	Not resolved	10-NOV-22 08.40.00.000000 AM	10-NOV-22 03.40.00.000000 PM	Confirmed
6000000009	-	google meets	partially resolved	24-DEC-22 08.30.00.000000 AM	24-DEC-22 03.30.00.000000 PM	pending

Download CSV
10 rows selected.

- After deleting:

SESSION_ID	SESSION_DESCRIPTION	SESSION_MEETING_METHOD	SESSION_OUTCOME	START_TIME_STAMP	END_TIME_STAMP	SESSION_STATUS
6000000002	How to implement/import graphic design in iOS apps	offline	Not resolved	07-JUL-22 03.04.02.000000 AM	07-JUL-22 05.04.02.000000 PM	Confirmed
6000000003	Video game animation format	google meets	Fully resolved	02-FEB-22 08.29.30.000000 AM	02-FEB-22 09.29.30.000000 AM	Completed
6000000004	-	Zoom	partially resolved	23-AUG-22 09.00.09.000000 PM	23-AUG-22 09.40.09.000000 PM	Confirmed
6000000006	-	google meets	Not resolved	19-SEP-22 01.00.00.000000 PM	19-SEP-22 01.40.00.000000 PM	Confirmed
6000000007	-	offline	partially resolved	23-OCT-22 09.30.45.000000 PM	23-OCT-22 10.30.45.000000 PM	pending
6000000008	-	Zoom	Not resolved	10-NOV-22 08.40.00.000000 AM	10-NOV-22 03.40.00.000000 PM	Confirmed
6000000009	-	google meets	partially resolved	24-DEC-22 08.30.00.000000 AM	24-DEC-22 03.30.00.000000 PM	pending

Download CSV
7 rows selected.

APPENDIX

- User_pf data:

USER_ID	USER_FIRST_NAME	USER_LAST_NAME	PHONE_NUMBER	EMAIL	USER_START_DATE
1000000000	Sara	Al Shareef	567343459	sarashareef@gmail.com	09-NOV-19
1000000001	Ahmed	Al Ghamdi	556729247	ahmedalghamdi@yahoo.com	18-MAR-20
1000000002	Noor	Al Rajhi	507827553	Nooralrajhi@outlook.com	20-DEC-20
1000000003	Sultan	Ba Zuhir	547534567	sultanbazuhair@gmail.com	14-JUN-21
1000000004	Judy	Al Garni	557462768	judyalgarni@outlook.com	21-FEB-22
1000000005	Jury	Mola	509748656	jurymola@hotmail.com	15-OCT-19
1000000006	Adam	koja	507366764	adamkoja@outlook.com	26-JUL-21
1000000007	Faris	Thanyan	543847746	faristhanyan@hotmail.com	30-JAN-22
1000000008	Reem	Sobhy	557547532	reemsobhy@gmail.com	24-FEB-21
1000000009	Rawan	Alyami	504674542	ranwaalyami@gmail.com	08-AUG-20
1000000010	layan	Shukor	558565389	layanshukor@yahoo.com	09-JUL-19
1000000011	Maan	Al dosory	562873694	maanaldosory@hotmail.com	23-SEP-20
1000000012	Hadeel	Al ghatani	508675355	hadeelalghatani@gmail.com	22-DEC-21
1000000013	rami	Al bany	509674686	ramialbany@gmail.com	27-NOV-19
1000000014	yousof	Al harbi	508743548	yousofalharbi@gmail.com	19-AUG-20
1000000015	Ahlam	Al Magrbi	568652977	ahalmalmagrbi@hotmail.com	20-FEB-20
1000000016	Salem	Al Salem	503586846	salemalsalem@yahoo.com	07-JUL-19
1000000017	rashid	Al thibiti	553896946	rashidalthibti@hotmail.com	08-SEP-21
1000000018	Shams	Al bargi	576257795	shamasalbargi@gmail.com	01-JAN-22
1000000019	Ward	akmal	505896986	wardakmal@gmail.com	02-JUN-21

Download CSV

20 rows selected.

- User_language data:

USER_ID	LANGUAGE
1000000000	Arabic
1000000000	English
1000000001	Arabic
1000000001	English
1000000001	Spain
1000000002	Arabic
1000000002	English
1000000003	English
1000000003	Spain
1000000004	Arabic
1000000004	English
1000000004	France
1000000005	English
1000000005	France
1000000006	Arabic
1000000006	English
1000000007	English
1000000008	Arabic
1000000008	English
1000000008	France
1000000009	Arabic
1000000010	English
1000000011	France
1000000012	Arabic
1000000013	Arabic
1000000014	English
1000000015	English
1000000016	English
1000000017	Spain
1000000018	Arabic
1000000018	English
1000000019	Arabic
1000000019	English

Download CSV

33 rows selected.

- Mentor data:

MENTOR_ID	YEARS_OF_EXPERIENCE	AVAILABILITY	IS_ACTIVE
1000000000	7	1	1
1000000001	8	0	1
1000000002	4	1	1
1000000003	4	0	0
1000000004	6	1	1
1000000005	5	1	0
1000000006	8	0	1
1000000007	5	1	1
1000000008	4	0	0
1000000009	6	1	1

Download CSV
10 rows selected.

- Mentor_meeting_method data:

MENTOR_ID	MENTOR_MEETING_METHOD
1000000000	Zoom
1000000001	Offline
1000000001	Zoom
1000000002	Offline
1000000003	Offline
1000000003	Google meets
1000000003	Zoom
1000000004	Zoom
1000000005	Offline
1000000006	Google meets
1000000006	Zoom
1000000007	Offline
1000000008	Zoom
1000000009	Google meets

Download CSV
14 rows selected.

- Mentee data:

MENTEE_ID	ROLE	MENTEE_SCORE	MEMBERSHIP_ID
1000000010	Student	80	7000000000
1000000011	Teacher	65	7000000001
1000000012	Employee	90	7000000002
1000000013	Employee	100	7000000003
1000000014	Student	50	7000000004
1000000015	Student	30	7000000005
1000000016	Student	80	7000000006
1000000017	Employee	80	7000000007
1000000018	Employee	80	7000000008
1000000019	Student	50	7000000009

Download CSV
10 rows selected.

- Session_ data:

SESSION_ID	SESSION_DESCRIPTION	SESSION_MEETING_METHOD	SESSION_OUTCOME	START_TIME_STAMP	END_TIME_STAMP	SESSION_STATUS
6000000000	-	Zoom	Fully resolved	04-APR-20 08.00.00.000000 PM	04-APR-20 08.40.00.000000 PM	Completed
6000000001	-	Zoom	partially resolved	31-DEC-19 10.30.00.000000 AM	31-DEC-20 03.30.00.000000 PM	Completed
6000000002	How to implement/import graphic design in iOS apps	offline	Not resolved	07-JUL-22 03.04.02.000000 AM	07-JUL-22 05.04.02.000000 PM	Confirmed
6000000003	Video game animation format	google meets	Fully resolved	02-FEB-22 08.29.30.000000 AM	02-FEB-22 09.29.30.000000 AM	Completed
6000000004	-	Zoom	partially resolved	23-AUG-22 09.00.09.000000 PM	23-AUG-22 09.40.09.000000 PM	Confirmed
6000000005	Running node.js in iOS/Android	offline	Fully resolved	23-FEB-21 08.00.08.000000 PM	23-FEB-21 08.40.08.000000 PM	Completed
6000000006	-	google meets	Not resolved	19-SEP-22 01.00.00.000000 PM	19-SEP-22 01.40.00.000000 PM	Confirmed
6000000007	-	offline	partially resolved	23-OCT-22 09.30.45.000000 PM	23-OCT-22 10.30.45.000000 PM	pending
6000000008	-	Zoom	Not resolved	10-NOV-22 08.40.00.000000 AM	10-NOV-22 03.40.00.000000 PM	Confirmed
6000000009	-	google meets	partially resolved	24-DEC-22 08.30.00.000000 AM	24-DEC-22 03.30.00.000000 PM	pending

Download CSV
10 rows selected.

- Session_Feild data:

SESSION_ID	FEILD_ID
6000000000	2000000000
6000000001	2000000001
6000000002	2000000002
6000000003	2000000003
6000000004	2000000005
6000000005	2000000005
6000000006	2000000005
6000000007	2000000007
6000000008	2000000008
6000000009	2000000009

Download CSV
10 rows selected.

- Membership data:

MEMBERSHIP_ID	TIER	MEMBERSHIP_START_DATE
7000000000	standard	09-JUL-19
7000000001	gold	25-SEP-20
7000000002	silver	22-DEC-21
7000000003	bronze	29-NOV-19
7000000004	standard	19-AUG-20
7000000005	standard	20-FEB-20
7000000006	standard	07-JUL-19
7000000007	bronze	19-SEP-21
7000000008	gold	01-JAN-22
7000000009	gold	02-JUN-21

Download CSV
10 rows selected.

- Tier_Cost data:

TIER	COST
standard	0
gold	15
silver	10
bronze	5

Download CSV
4 rows selected.

- Post_Comment data:

COMMENT_ID	POST_ID
8000000010	5000000000
8000000011	5000000001
8000000012	5000000002
8000000013	5000000003
8000000014	5000000004
8000000015	5000000005
8000000016	5000000006
8000000017	5000000007
8000000018	5000000008
8000000019	5000000009

Download CSV
10 rows selected.

- Blog data:

BLOG_ID	BLOG_NAME	BLOG_DISCRIPTION	MRG_ID
4000000000	iOS Development	In my blog you will find posts that talk about IOS Development	1000000000
4000000001	Back-End Development articles	If you interested in Back-End Development, you will find an article about it in my blog	1000000001
4000000002	Full stack dev	-	1000000002
4000000003	All about game developers	All about graphic design	1000000003
4000000004	iOS and Android Development	Step by step to be a game developers	1000000004
4000000005	Understanding application developer skills	-	1000000005
4000000006	Android Development	developing and modifying source code for software applications	1000000006
4000000007	About network	-	1000000007
4000000008	Front-End Development articles	Everything about network	1000000008
4000000009	About graphic design	-	1000000009

Download CSV
10 rows selected.

- Blog_Field data:

BLOG_ID	FIELD_ID
4000000000	2000000000
4000000001	2000000001
4000000002	2000000002
4000000003	2000000003
4000000004	2000000004
4000000005	2000000005
4000000006	2000000006
4000000007	2000000007
4000000008	2000000008
4000000009	2000000009

Download CSV
10 rows selected.

- Post data:

POST_ID	POST_NAME	POST_DISCRIPTION	BLOG_ID
5000000010	The 8 Types of Graphic Design You Need To Know	About The 8 fundamental types of graphic design	4000000003
5000000011	What is graphic design	Simple introduction about graphic design for beginners	4000000003
5000000012	How video game are made	-	4000000004
5000000013	The power of making game	-	4000000004
5000000014	How video game development has changed over the last decade	-	4000000004
5000000015	Difference Between Front End Development and Back End Development	-	4000000002
5000000016	5 Steps to Building and Operating an Effective Security Operations Center	-	4000000008
5000000017	Packet Switching Methods on Cisco Networks	-	4000000008
5000000018	The Art of Network Architecture: Applying Modularity	about Hierarchical Design	4000000008
5000000019	VLAN Communications: Making Networks Talk to Each Other	important VLAN concepts	4000000008
5000000020	The 8 Types of Graphic Design You Need To Know	About The 8 fundamental types of graphic design	4000000003
5000000021	What is graphic design	Simple introduction about graphic design for beginners	4000000003
5000000022	How video game are made	-	4000000004
5000000023	The power of making game	-	4000000004
5000000024	How video game development has changed over the last decade	-	4000000004
5000000025	Difference Between Front End Development and Back End Development	-	4000000002
5000000026	5 Steps to Building and Operating an Effective Security Operations Center	-	4000000008
5000000027	Packet Switching Methods on Cisco Networks	-	4000000008
5000000028	The Art of Network Architecture: Applying Modularity	about Hierarchical Design	4000000008
5000000029	VLAN Communications: Making Networks Talk to Each Other	important VLAN concepts	4000000008
5000000000	The 8 Types of Graphic Design You Need To Know	About The 8 fundamental types of graphic design	4000000003
5000000001	What is graphic design	Simple introduction about graphic design for beginners	4000000003
5000000002	How video game are made	-	4000000004
5000000003	The power of making game	-	4000000004
5000000004	How video game development has changed over the last decade	-	4000000004
5000000005	Difference Between Front End Development and Back End Development	-	4000000002
5000000006	5 Steps to Building and Operating an Effective Security Operations Center	-	4000000008
5000000007	Packet Switching Methods on Cisco Networks	-	4000000008
5000000008	The Art of Network Architecture: Applying Modularity	about Hierarchical Design	4000000008
5000000009	VLAN Communications: Making Networks Talk to Each Other	important VLAN concepts	4000000008

Download CSV

30 rows selected.

- Post_Field data:

POST_ID	FIELD_ID
5000000000	2000000003
5000000001	2000000003
5000000002	2000000004
5000000003	2000000004
5000000004	2000000004
5000000005	2000000002
5000000006	2000000008
5000000007	2000000008
5000000008	2000000008
5000000009	2000000008

Download CSV

10 rows selected.

- Session_Mentee_Mentor data:

SESSION_ID	MENTEE_ID	MENTOR_ID
6000000000	1000000010	1000000000
6000000001	1000000011	1000000001
6000000002	1000000012	1000000002
6000000003	1000000013	1000000003
6000000004	1000000014	1000000005
6000000005	1000000014	1000000005
6000000006	1000000016	1000000005
6000000007	1000000017	1000000007
6000000008	1000000018	1000000008
6000000009	1000000019	1000000009

Download CSV

10 rows selected.

- User_Field data:

USER_ID	FIELD_ID
1000000000	2000000000
1000000001	2000000001
1000000002	2000000002
1000000003	2000000003
1000000004	2000000004
1000000005	2000000005
1000000006	2000000006
1000000007	2000000007
1000000008	2000000008
1000000009	2000000009
1000000010	2000000001
1000000010	2000000008
1000000011	2000000007
1000000011	2000000008
1000000012	2000000002
1000000013	2000000002
1000000014	2000000000
1000000015	2000000009
1000000016	2000000009
1000000017	2000000004
1000000018	2000000004
1000000019	2000000006

- Field data:

FIELD_ID	FIELD_DESCRIPTION
2000000000	iOS (formerly iPhone OS) is a mobile operating system created and developed by Apple Inc. exclusively for its hardware
2000000001	The back- end of a website consists of a server, an application, and a database A back-end developer builds and maintains the technology that powers those components which, together, enable the user-facing side of the website to even exist in the first place.
2000000002	In software architecture, there may be many layers between the hardware and end user. The front is an abstraction, simplifying the underlying component by providing a user-friendly interface, while the back usually handles data storage and business logic. In telecommunication, the front can be considered a device or service, while the back is the infrastructure that supports provision of service. A rule of thumb is that the client-side (or "frontend") is any component manipulated by the user. The server-side (or "backend") code usually resides on the server, often far removed physically from the user.
2000000003	graphic designers working in user experience (UX) design must justify stylistic choices regarding, say, image locations and font with a human-centered approach
2000000004	Video game developers help transform games from a concept to a playable reality. They do this by coding visual elements, programming features, and testing iterations until a game is ready for market.
2000000005	Googles Android and Apple iOS are operating systems used primarily in mobile technology, such as smartphones and tablets
2000000006	Application development is the process of designing, building, and implementing software applications. It can be done by massive organizations with large teams working on projects, or by a single freelance developer. Application development defines the process of how the application is made
2000000007	Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.
2000000008	A network is a collection of computers, servers, mainframes, network devices, peripherals, or otherr devices connected to allow data sharing
2000000009	The front-end of a website is the part that users interact with. Everything that you see when you're navigating around the Internet

- Field_Topic data:

FIELD_ID	TOPIC
2000000005	Android
2000000007	Android
2000000006	Application-Development
2000000001	Back-End
2000000002	Back-End
2000000002	Front-End
2000000009	Front-End
2000000003	Graphic-Design
2000000000	IOS
2000000005	IOS
2000000008	Network
2000000004	Video-Game-Devlopment

- Discussion_Room data:

DISCUSSION_ROOM_ID	DISCUSSION_ROOM_NAME	DISCUSSION_ROOM_DESCRIPTION	OUTCOME	DISCUSSION_ROOM_START_DATE	STATUS	MENTEE_ID
3000000000	How to open documents folder in iOS swift	-	Canceled	01-FEB-20	Not resolved	1000000010
3000000001	What is a good back end to use with AngularJS	-	Completed	17-APR-19	Fully resolved	1000000011
3000000002	Separate back end and front-end apps on same domain	-	Completed	17-APR-19	Fully resolved	1000000012
3000000003	Graphic design program is not working	-	-	08-AUG-22	partially resolved	1000000013
3000000004	Issue with flickering in Java 2D Video Game	-	Confirmed	09-DEC-22	partially resolved	1000000014
3000000005	Detecting iOS / Android Operating system	I did some research, and this question came up, but not the way I intended. I am creating a page for the client, which is a QR code, a place to download the app. So, it is not necessary to print many QR codes on the page, I would like to detect the current OS (Apple / Android / Other [not supported]) and modify my stuff based on this value.	Confirmed	06-JAN-22	Not resolved	1000000015
3000000006	How to structure an application?	-	Completed	01-JAN-22	Fully resolved	1000000016
3000000007	Android Studio Flutter Android App issues with Work Manager	-	Canceled	02-OCT-20	Not resolved	1000000017
3000000008	How to use Chrome network debugger with redirects	-	-	10-NOV-22	Not resolved	1000000018
3000000009	what is `optimistic updates` in front-end development	-	Completed	01-APR-22	partially resolved	1000000019

- Discussion_Room_Field data:

DISCUSSION_ROOM_ID	FIELD_ID
3000000000	2000000000
3000000001	2000000001
3000000002	2000000002
3000000002	2000000007
3000000003	2000000003
3000000004	2000000004
3000000005	2000000005
3000000006	2000000006
3000000007	2000000007
3000000008	2000000008
3000000009	2000000009

- Discussion_Room_Comment data:

COMMENT_ID	DISCUSSION_ROOM_ID
8000000000	3000000000
8000000001	3000000001
8000000002	3000000002
8000000003	3000000003
8000000004	3000000004
8000000005	3000000005
8000000006	3000000006
8000000007	3000000007
8000000008	3000000008
8000000009	3000000009

Download CSV
10 rows selected.

- Comment_ data:

COMMENT_ID	AUTHORID	CONTENT	CREATION_DATE	UPDATE_DATE	UPVOTE
8000000000	1000000011	My issue still not fixed	02-FEB-20	02-FEB-20	3
8000000001	1000000009	This was a great explanation	19-APR-19	22-APR-19	0
8000000002	1000000017	Thank you so much!!	23-DEC-21	27-DEC-21	6
8000000003	1000000008	amazing	18-DEC-21	18-DEC-21	2
8000000004	1000000006	Good, but could have been better	08-AUG-22	09-AUG-22	2
8000000005	1000000020	Well, part of the problem is solved	09-DEC-22	11-DEC-22	0
8000000006	1000000008	My problem is not solved	01-JUN-22	01-JUN-22	3
8000000007	1000000003	My problem is completely resolved!	01-JAN-22	03-JAN-22	4
8000000008	1000000015	This did not work	09-OCT-21	10-OCT-21	0
8000000009	1000000004	Well, I still need more help	10-NOV-22	11-NOV-22	0
8000000010	1000000003	Amazing!!	22-JAN-21	22-FEB-21	4
8000000011	1000000011	This is cool	24-APR-20	25-APR-20	2
8000000012	1000000013	Great	03-APR-21	04-APR-21	8
8000000013	1000000006	Thanks	17-AUG-19	17-AUG-19	10
8000000014	1000000012	Brilliant	24-JUL-21	26-JUL-21	0
8000000015	1000000008	This was useful	04-APR-22	04-APR-22	7
8000000016	1000000005	Very smart	23-JUL-21	24-JUL-21	9
8000000017	1000000000	fairly good	29-JAN-19	02-FEB-19	1
8000000018	1000000014	Not bad	20-DEC-20	20-DEC-20	0
8000000019	1000000013	Wow!!	12-DEC-22	12-DEC-22	3
8000000020	1000000016	That was extremely useful	04-APR-20	14-NOV-21	6
8000000021	1000000015	I still need more help	29-JAN-20	29-JAN-20	9
8000000022	1000000001	Not that much help	07-JUL-22	07-JUL-22	0
8000000023	1000000002	Fabulous	02-FEB-20	04-FEB-20	3
8000000024	1000000014	My problem is still there	23-AUG-22	24-AUG-22	4
8000000025	1000000008	This was a successful solution	23-FEB-21	24-FEB-21	5
8000000026	1000000019	Nothing worked	19-SEP-22	20-SEP-22	6
8000000027	1000000009	Did not get much use	23-OCT-22	25-DEC-22	7
8000000028	1000000014	No useful solutions	10-NOV-22	11-NOV-22	9
8000000029	1000000013	Good but not perfect	24-DEC-22	27-DEC-22	0

Download CSV
30 rows selected.

- Review_Comment:

COMMENT_ID	REVIEW_ID
8000000020	9000000000
8000000021	9000000001
8000000022	9000000002
8000000023	9000000003
8000000024	9000000004
8000000025	9000000005
8000000026	9000000006
8000000027	9000000007
8000000028	9000000008
8000000029	9000000009

Download CSV
10 rows selected.

- Session_Review data:

SESSION_ID	REVIEW_ID
6000000000	9000000000
6000000001	9000000001
6000000002	9000000002
6000000003	9000000003
6000000004	9000000004
6000000005	9000000005
6000000006	9000000006
6000000007	9000000007
6000000008	9000000008
6000000009	9000000009

Download CSV
10 rows selected.

- Review data:

SESSION_ID	REVIEW_ID	IS_ANONYMOUS	SCORE
6000000000	9000000000	0	5
6000000001	9000000001	1	3
6000000002	9000000002	0	1
6000000003	9000000003	0	4
6000000004	9000000004	1	2
6000000005	9000000005	1	5
6000000006	9000000006	1	1
6000000007	9000000007	0	3
6000000008	9000000008	1	1
6000000009	9000000009	1	2

Download CSV

10 rows selected.