CRM Database Documentation

Table of Contents

[**1** **Requirement Analysis** 6](#_Toc184984722)

[1.1 Brief Introduction 6](#_Toc184984723)

[**1.1.1** **Brief Explanation of The Requirements** 6](#_Toc184984724)

[**1.1.2** **Requirements Analysis** 7](#_Toc184984725)

[1.1.3 Resources used 7](#_Toc184984726)

[1.2 List of data fields (Entities and their attributes) 8](#_Toc184984727)

[**1. Members and Related Tables** 8](#_Toc184984728)

[Members 8](#_Toc184984729)

[Member Address 8](#_Toc184984730)

[Memberships 8](#_Toc184984731)

[Member Tags 8](#_Toc184984732)

[Member Interests 9](#_Toc184984733)

[Engagements (Derived Information) 9](#_Toc184984734)

[Subscriptions 9](#_Toc184984735)

[Feedback 9](#_Toc184984736)

[Attendance Log 9](#_Toc184984737)

[Log-in Records 10](#_Toc184984738)

[**2. Events and Related Tables** 10](#_Toc184984739)

[Events 10](#_Toc184984740)

[Event Types 10](#_Toc184984741)

[Spaces 10](#_Toc184984742)

[Bookings 11](#_Toc184984743)

[**3. Payments and Related Tables** 11](#_Toc184984744)

[Payments 11](#_Toc184984745)

[Payment Reasons 11](#_Toc184984746)

[**4. Inquiries and Related Tables** 11](#_Toc184984747)

[Inquiries 11](#_Toc184984748)

[Inquiry Types 12](#_Toc184984749)

[Inquiry User Types 12](#_Toc184984750)

[**5. Tags and Interests** 12](#_Toc184984751)

[Tags 12](#_Toc184984752)

[Interests 12](#_Toc184984753)

[**6. Staff and Related Tables** 12](#_Toc184984754)

[Staff 12](#_Toc184984755)

[Staff Address 13](#_Toc184984756)

[Staff Attendance 13](#_Toc184984757)

[Staff Clearance 13](#_Toc184984758)

[Staff Salary Type 13](#_Toc184984759)

[Workplace 13](#_Toc184984760)

[Role 13](#_Toc184984761)

[**7. Founding Members and Related Tables** 14](#_Toc184984762)

[Founding Members 14](#_Toc184984763)

[Founding Member Address 14](#_Toc184984764)

[**2** **Database design** 15](#_Toc184984765)

[**2.1 Entity Relationship Modelling** 15](#_Toc184984766)

[**2.1.1** **Initial Entity Relationship Model** 15](#_Toc184984767)

[**2.1.2** **Extended Entity Relationship Model** 17](#_Toc184984768)

[2.2 Normalised Model 19](#_Toc184984769)

[**2.3 Database Schema** 22](#_Toc184984770)

[**3.** **Mapping** 30](#_Toc184984771)

[3.1 General Usage Patterns 30](#_Toc184984772)

[3.2 Individual Member Usage 31](#_Toc184984773)

[3.3 Event Participation and Interests 32](#_Toc184984774)

[3.4 Audience Segmentation and Personalization 33](#_Toc184984775)

[3.5 Space Utilisation and Capacity Planning 34](#_Toc184984776)

[3.6 Reporting and Data Integration 35](#_Toc184984777)

[3.7 User journey and Acquisition 36](#_Toc184984778)

[**6.** **SQL Queries** 56](#_Toc184984779)

[5.1 Query 1 56](#_Toc184984780)

[**5.1.2**  **Query in natural language (Question from the list provided on Canvas)** 56](#_Toc184984781)

[**5.1.3**  **SQL Code and output** 56](#_Toc184984782)

[**5.1.4**  **Explain the output of the data (was this what was predicted?)** 58](#_Toc184984783)

[5.2 Query 2 58](#_Toc184984784)

[**5.2.2** **Query in natural language (Question from the list provided on Canvas)** 58](#_Toc184984785)

[**5.2.3** **SQL Code and output** 58](#_Toc184984786)

[**5.2.4** **Explain the output of the data (was this what was predicted?)** 59](#_Toc184984787)

[5.3 Query 3 59](#_Toc184984788)

[**5.3.2**  **Query in natural language (Question from the list provided on Canvas)** 59](#_Toc184984789)

[**5.3.3**  **SQL Code and output** 60](#_Toc184984790)

[**5.3.4**   **Explain the output of the data (was this what was predicted?)** 60](#_Toc184984791)

[5.4 Query 4 60](#_Toc184984792)

[**5.4.2**  **Query in natural language (Question from the list provided on Canvas)** 60](#_Toc184984793)

[**5.4.3**  **SQL Code and output** 60](#_Toc184984794)

[**5.4.4**  **Explain the output of the data (was this what was predicted?)** 62](#_Toc184984795)

[5.5 Query 5 62](#_Toc184984796)

[**5.5.2**  **Query in natural language (Question from the list provided on Canvas)** 62](#_Toc184984797)

[**5.5.3**  **SQL Code and output** 62](#_Toc184984798)

[**5.5.4**  **Explain the output of the data (was this what was predicted?)** 64](#_Toc184984799)

[5.6 Query 6 64](#_Toc184984800)

[**5.6.2**  **Query in natural language (Question from the list provided on Canvas)** 64](#_Toc184984801)

[**5.6.3**  **SQL Code and output** 64](#_Toc184984802)

[**5.6.4**  **Explain the output of the data (was this what was predicted?)** 65](#_Toc184984803)

[5.7 Query 7 66](#_Toc184984804)

[**5.7.2**  **Query in natural language (Question from the list provided on Canvas)** 66](#_Toc184984805)

[**5.7.3**  **SQL Code and output** 66](#_Toc184984806)

[**5.7.4**  **Explain the output of the data (was this what was predicted?)** 67](#_Toc184984807)

[5.8 Query 8 67](#_Toc184984808)

[**5.8.2**  **Query in natural language (Question from the list provided on Canvas)** 67](#_Toc184984809)

[**5.8.4**  **Explain the output of the data (was this what was predicted?)** 69](#_Toc184984810)

[5.9 Query 9 69](#_Toc184984811)

[**5.9.1**  **For what purpose will this query be used** 69](#_Toc184984812)

[**5.9.2**  **Query in natural language** 69](#_Toc184984813)

[**5.9.3**  **SQL Code and output** 69](#_Toc184984814)

[**5.9.4**  **Explain the output of the data (was this what was predicted?)** 70](#_Toc184984815)

[5.10 Query 10 70](#_Toc184984816)

[**5.10.1 For what purpose will this query be used** 70](#_Toc184984817)

[**5.10.2 Query in natural language** 71](#_Toc184984818)

[**5.10.3 SQL Code and output** 71](#_Toc184984819)

[**5.10.4 Explain the output of the data (was this what was predicted?)** 71](#_Toc184984820)

Database Report

# **1** **Requirement Analysis**

## 1.1 Brief Introduction

Together Culture provided us with a particular set of project requirements for our target project: a **database for CRM**. This database is intended to streamline **customer relationship management** by solving specific questions and requirements using SQL queries.

The project involves designing and implementing an efficient relational database schema to store and manage a variety of customer data. The deliverables include a set of SQL queries that address the provided requirements, ensuring accurate data retrieval, updates, and management. The solution must prioritize robust error handling to ensure reliability and usability. Additionally, the database should be designed to function seamlessly across different platforms, maintaining compatibility with standard SQL environments.

### **Brief Explanation of The Requirements**

(T. C., 2024) Together Culture is a Community Interest Company (CIC) focused on building a more equitable and ecological creative economy. Their aim is to transition from a manual customer data management system to a digital Customer Relationship Management (CRM) solution to:

* Enable the organization to manage members
* Track their activities.
* Monitor their engagement with events and contents.
* Analyse patterns that could help improve the community experience.

This aims to provide valuable insights into members’ behaviour and help take decisions related to content delivery, event planning, and resource allocation. Additionally, the system will streamline administrative tasks, improve operational efficiency, and enhance the overall member experience.

Moreover, this project shows the database, and its components used for the CRM needed.

### **Requirements Analysis**

**Functional requirements:**

* Mine data on community funnel progression and journey analytics.
* Tag and categorize user activities to track trends in a relational database.
* Add new members and record their interests.

**Non-functional requirements:**

* Ensure user data is stored securely and compliant with GDPR/data protection standards.
* Fast search and data retrieval for both members and administrators.
* Reliance on relational databases for tagging and tracking data.

### **1.1.3 Resources used**

* Opera Web Browser 115.0.5322.58 for Desktop
* Microsoft 365
* MySQL Workbench 8.0.38 Community
* Lucid chart
* Device Details:
  + 1. Edition Windows 11 Home Single Language
    2. Version 23H2
    3. OS build 22631.4541
    4. Processor 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz 3.11 GHz
    5. Installed RAM 8.00 GB (7.79 GB usable)
    6. System type 64-bit operating systeSm, x64-based processo

## 1.2 List of data fields (Entities and their attributes)

### **1. Members and Related Tables**

#### Members

* **Member ID (PK):** Unique identifier for each member.
* **First Name:** Member's first name.
* **Last Name:** Member's last name.
* **Email:** Member's email address.
* **Phone Number:** Member's phone number.
* **Membership ID (FK):** Foreign key linking to the Membership table.
* **Password:** Member's password for logging into the system.
* **Joining Date:** The date the member joined.
* **Subscribed to Newsletter:** Indicates whether the member has subscribed to newsletters.

#### Member Address

* **Address ID (PK):** Unique identifier for each address.
* **Member ID (FK):** Foreign key linking to the Members table.
* **House Number:** House or apartment number.
* **Street:** Street name of the member's address.
* **City:** City where the member resides.
* **State:** State where the member resides.
* **City Code:** Postal or ZIP code.
* **Country:** Country where the member resides.

#### Memberships

* **Membership ID (PK):** Unique identifier for each membership.
* **Type:** Type of membership (e.g., Basic, Premium).
* **Price:** Cost of the membership.

#### Member Tags

* **Member ID (FK):** Foreign key linking to the Members table.
* **Tag ID (FK):** Foreign key linking to the Tags table.

#### Member Interests

* **Interest ID (FK):** Foreign key linking to the Interests table.
* **Member ID (FK):** Foreign key linking to the Members table.

#### Engagements (Derived Information)

* **Member ID (FK):** Foreign key linking to the Members table.
* **Number of Sign-ins:** Count of member's log-ins.
* **Events Booked:** Count of events booked by the member.
* **Events Attended:** Count of events attended by the member.

#### Subscriptions

* **Subscription ID (PK):** Unique identifier for each subscription.
* **Payment ID (FK):** Foreign key linking to the Payments table.
* **Member ID (FK):** Foreign key linking to the Members table.
* **Membership ID (FK):** Foreign key linking to the Memberships table.
* **Start Date:** Subscription start date.
* **End Date:** Subscription end date.

#### Feedback

* **Feedback ID (PK):** Unique identifier for each feedback.
* **Member ID (FK):** Foreign key linking to the Members table.
* **Description:** Textual feedback provided by the member.

#### Attendance Log

* **Attendance ID (PK):** Unique identifier for each attendance record.
* **Member ID (FK):** Foreign key linking to the Members table.
* **Attendee Name:** Name of the attendee.
* **Email:** Email of the attendee.
* **Event ID (FK):** Foreign key linking to the Events table.
* **Space ID (FK):** Foreign key linking to the Spaces table.
* **Check-in Time:** Timestamp of when the member checked in.
* **Check-in Date:** Date of when Member checks in.

#### Log-in Records

* **Log-in ID (PK):** Unique identifier for each log-in record.
* **Member ID (FK):** Foreign key linking to the Members table.
* **Log-in Time:** Timestamp of the log-in.
* **Log-in Date:** Date of Log-in
* **Log-in Device:** Device used for logging in.

### **2. Events and Related Tables**

#### Events

* **Event ID (PK):** Unique identifier for each event.
* **Name:** Name of the event.
* **Start Time:** Starting time of event.
* **End Time:** End time of event.
* **Date:** Date when the event is scheduled to occur.
* **Event Type ID (FK):** Foreign key linking to the Event Types table.
* **Space ID (FK):** Foreign key linking to the Spaces table.
* **Expected Attendance:** The number of attendees expected for the event.
* **Space Hire Type:** Specifies the type of space hire (e.g., Full Access, Partial Access).
* **Public Price:** Price for non-members to attend the event.
* **Member Price:** Discounted price for members to attend the event.

#### Event Types

* **Event Type ID (PK):** Unique identifier for each event type.
* **Description:** Describes the type of event (e.g., Workshop, Seminar, Party).

#### Spaces

* **Space ID (PK):** Unique identifier for each space.
* **Name:** Name of the space (e.g., Conference Room A, Hall 1).
* **Capacity:** Maximum number of people the space can accommodate.

Bookings

* **Booking ID (PK):** Unique identifier for each booking.
* **Payment ID (FK):** Foreign key linking to the Payments table.
* **Member ID (FK):** Foreign key linking to the Members table.
* **Event ID (FK):** Foreign key linking to the Events table.

### **3. Payments and Related Tables**

#### Payments

* **Payment ID (PK):** Unique identifier for each payment.
* **Payment Reason ID (FK):** Foreign key linking to the Payment Reasons table.
* **Payment Method:** Method used for the payment (e.g., Credit Card, Cash, Bank Transfer).
* **Payment Amount:** Total amount of the payment.
* **Payment Date:** The date the payment was made.

#### Payment Reasons

* **Payment Reason ID (PK):** Unique identifier for each payment reason.
* **Description:** Description of the payment reason (e.g., Membership Fee, Event Booking).

### **4. Inquiries and Related Tables**

#### Inquiries

* **Inquiry ID (PK):** Unique identifier for each inquiry.
* **Member ID (FK):** Foreign key linking to the Members table.
* **Inquiry Type ID (FK):** Foreign key linking to the Inquiry Types table.
* **User Type ID (FK):** Foreign key linking to the Inquiry User Types table.
* **Inquiry Date:** Date when the inquiry was initiated.
* **Follow-up Count:** The number of follow-ups conducted for the inquiry.
* **Status:** Status of the inquiry (e.g., Open, Resolved).

#### Inquiry Types

* **Inquiry Type ID (PK):** Unique identifier for each inquiry type.
* **Description:** Description of the inquiry type (e.g., Complaint, Feedback, Information Request).

#### Inquiry User Types

* **User Type ID (PK):** Unique identifier for each user type.
* **Description:** Description of the user type (e.g., Member, Non-member).

### **5. Tags and Interests**

#### Tags

* **Tag ID (PK):** Unique identifier for each tag.
* **Name:** Name of the tag (e.g., Tech, Space, Law).

#### Interests

* **Interest ID (PK):** Unique identifier for each interest.
* **Name:** Name of the interest (e.g., Sharing Department, Caring Department).

### **6. Staff and Related Tables**

#### Staff

* **Staff ID (PK):** Unique identifier for each staff member.
* **First Name:** Staff member's first name.
* **Last Name:** Staff member's last name.
* **Role ID:** Role of the staff member (e.g., Manager, Technician) ID.
* **Workplace ID:** Type of workplace (e.g., Onsite, Remote) ID.
* **Contract ID:** Employment contract type (e.g., Full-time, Part-time) ID.
* **Staff Clearance ID (FK):** Foreign key linking to the Staff Clearance table.
* **Salary:** Salary of the staff members.
* **Salary Type ID:** Type of salary (e.g., Hourly, Monthly) ID.
* **Notice Period:** Duration of the notice period for resignation.
* **Date Joined:** The date the staff member joined the organization.

#### Staff Address

* **Staff Address ID (PK):** Unique identifier for each staff address.
* **Staff ID (FK):** Foreign key linking to the Staff table.
* **House Number:** House or apartment number.
* **Street:** Street name of the staff member's address.
* **City:** City where the staff member resides.
* **State:** State where the staff member resides.
* **City Code:** Postal or ZIP code.
* **Country:** Country where the staff member resides.

#### Staff Attendance

* **Staff Attendance ID (PK):** Unique identifier for each staff attendance record.
* **Staff ID (FK):** Foreign key linking to the Staff table.
* **Check-in Time:** Timestamp of when the staff member checked in.
* **Check-out Time:** Timestamp of when the staff member checked out.
* **Work Date:** The date of the work shift.

#### Staff Clearance

* **Staff Clearance ID (PK):** Unique identifier for each clearance level.
* **Description:** Description of the clearance level (e.g., Confidential, Restricted).

#### Staff Salary Type

* **Salary Type ID (PK):** Unique identifier for each salary type.
* **Description:** Describes the type of salary (e.g., Hourly, Monthly, Yearly).

#### Workplace

* **Workplace ID (PK):** Unique identifier for each workplace type.
* **Description:** Describes the workplace type (e.g., Remote, Onsite, Hybrid).

#### Role

* **Role ID (PK):** Unique identifier for each role.
* **Description:** Description of the role (e.g., Manager, Developer).

### **7. Founding Members and Related Tables**

#### Founding Members

* **Founding Member ID (PK):** Unique identifier for each founding member.
* **First Name:** Founding member's first name.
* **Last Name:** Founding member's last name.
* **Email:** Founding member's email address.
* **Phone Number:** Founding member's phone number.
* **Founding Date:** The date when the founding member officially joined.
* **Contribution:** The contributions (monetary or otherwise) made by the founding member.

#### Founding Member Address

* **Founding Member Address ID (PK):** Unique identifier for each founding member address.
* **Founding Member ID (FK):** Foreign key linking to the Founding Members table.
* **House Number:** House or apartment number.
* **Street:** Street name of the founding member's address.
* **City:** City where the founding member resides.
* **State:** State where the founding member resides.
* **City Code:** Postal or ZIP code.
* **Country:** Country where the founding member resides.

# **2** **Database design**

The following sections explain the progression of our database design through various diagrams, shoeing all kinds of changes and iterations it went through

## **2.1 Entity Relationship Modelling**

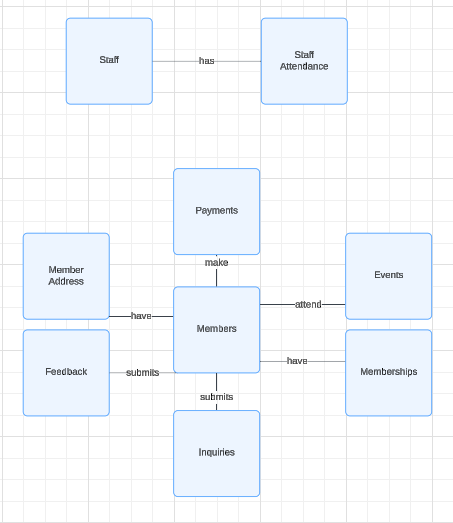
Here, three models of our database are presented:

* the initial ERD, which shows the tables which we came up with during the initial designing and brainstorming stage of this project,
* an EERD which includes detailed notations like relationship types, cardinalities, primary and foreign keys
* the final Normalized model, which has been refined using normalization techniques to produce the final EERD.

(*ER diagrams vs. EER diagrams: What’s the difference?*, 2018)

### **2.1.1** **Initial Entity Relationship Model**

* **Entities:** Tables initially added, are represented as rectangles.
* **Relationships:** Simple relationships explained through words like “make”, “attend” etc, with respect to Members/Staff and the actions they could make or entities they relate to.



This model doesn't show detailed cardinalities (like 1-to-many or many-to-many) or foreign keys yet

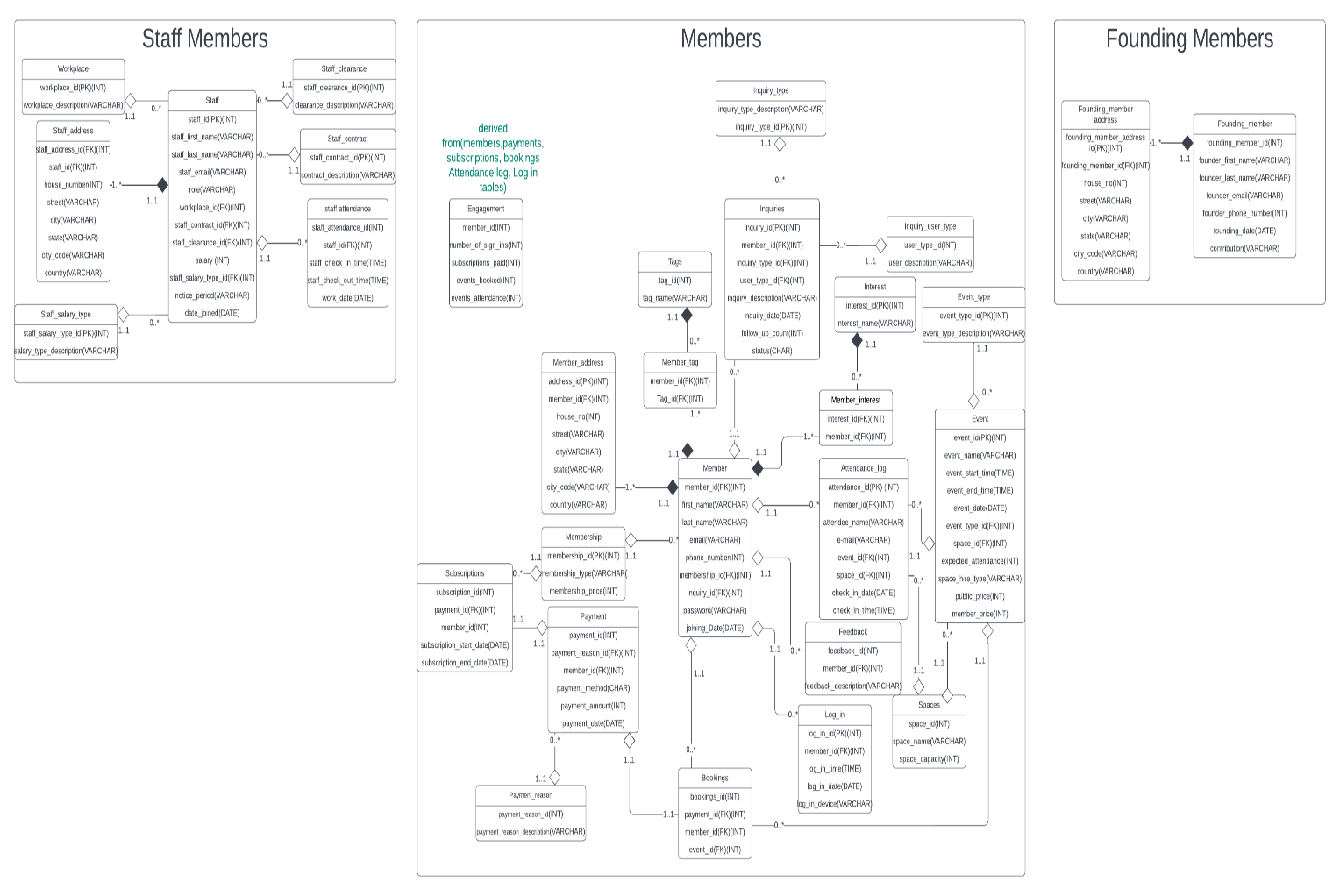
### **2.1.2** **Extended Entity Relationship Model**

In the **Extended ERD (EERD)**, we go beyond the basic representation and include the following:

* **Type and Cardinality of Relationships**: Here, we specify whether a relationship is **one-to-one**, **one-to-many**, or **many-to-many**.
* **Primary Keys (PK)** and **Foreign Keys (FK)**: Each entity's primary key is clearly indicated, and the foreign keys in the related entities are also shown to establish how entities are linked.
* Relationship lines depicting composition (with dark diamond) and aggregation (with light diamond), connecting entities are drawn with for clarity.

Changes to the Structure:

* The database now has additional number of tables to further expand the members and staff sections with tables like attendance log, log in, subscription, bookings, interests etc, on the members side, and tables like workplace, roles etc, on the staff side, with an addition of two tables to store Founding members information if needed.
* The Engagement section is not a table, but depicts the information which could be derived from other tables to get important user information in the future,



2.2 Normalised Model

Database normalization is a design principle aimed at organizing data efficiently to reduce redundancy, improve consistency, and maintain data integrity. It prevents issues related to insertion, deletion, and updates. (Chris, 2022)

**First Normal Form (1NF):**

A table is in 1NF if:

* + Each cell contains only atomic (indivisible) values.
  + The table has a primary key for identification.
  + There are no duplicate rows or columns. (Chris, 2022)

*In the current EERD*:

All tables satisfy 1NF, with unique primary keys for entities (e.g., member\_id, tag\_id). For linking tables like Member\_Tag, the composite keys (e.g., member\_id, tag\_id) act as the primary keys.

**Second Normal Form (2NF):**

A table is in 2NF if:

* + It is in 1NF.
  + All non-key attributes are fully dependent on the entire primary key, eliminating partial dependencies. (Chris, 2022)

*In the current EERD*:

All non-key attributes depend entirely on their respective primary keys, ensuring compliance with 2NF.

**Third Normal Form (3NF):**

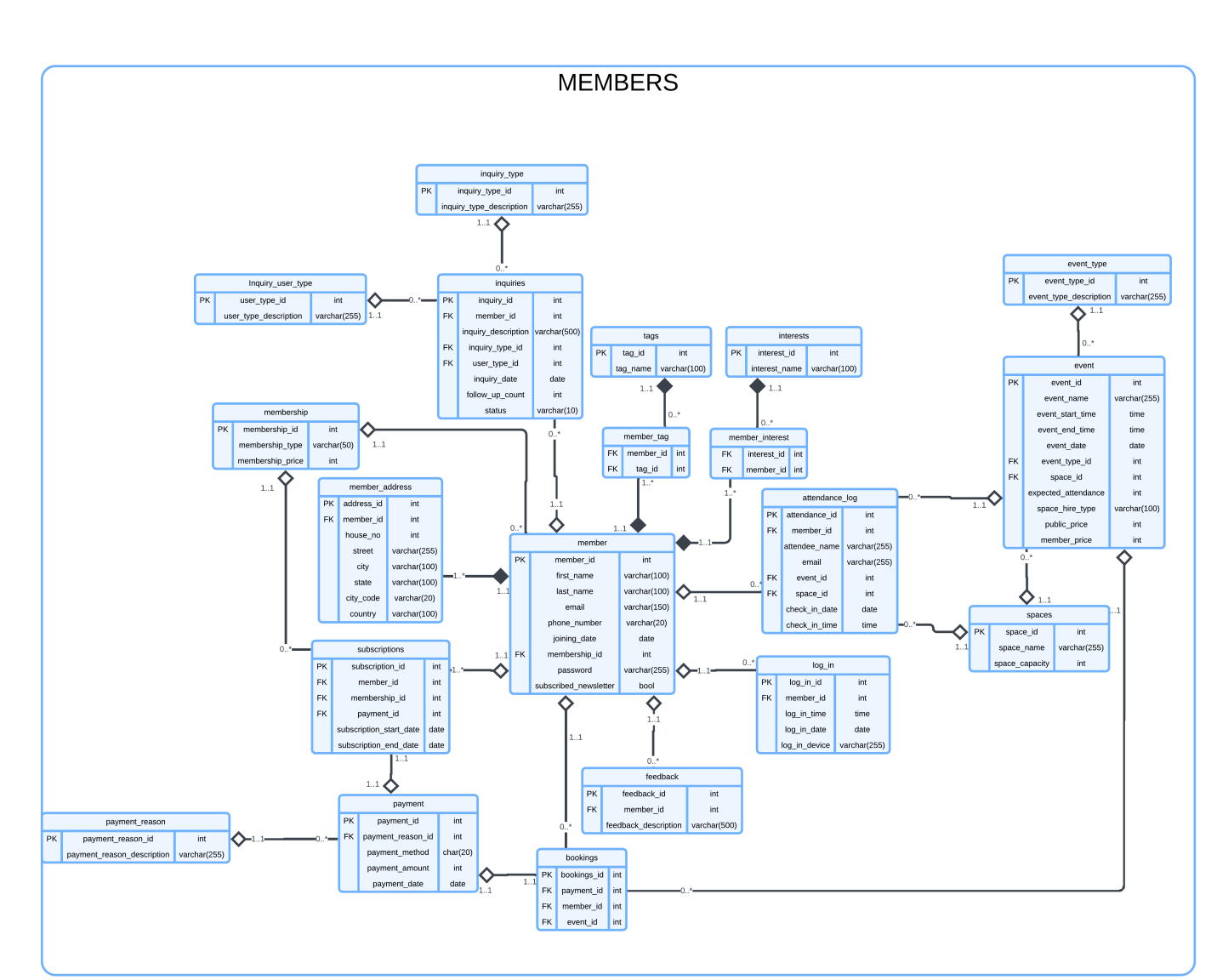
A table is in 3NF if:

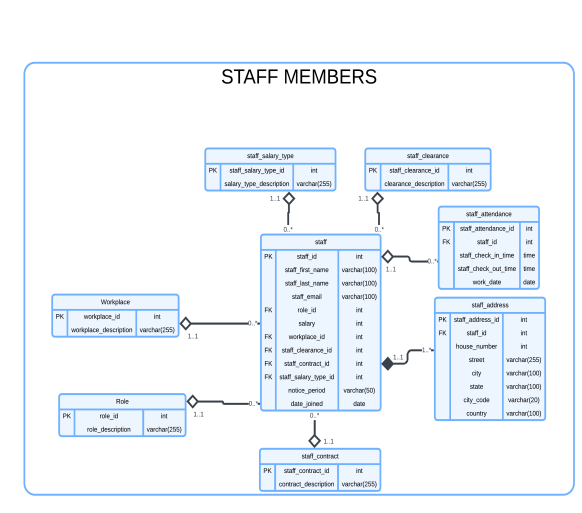
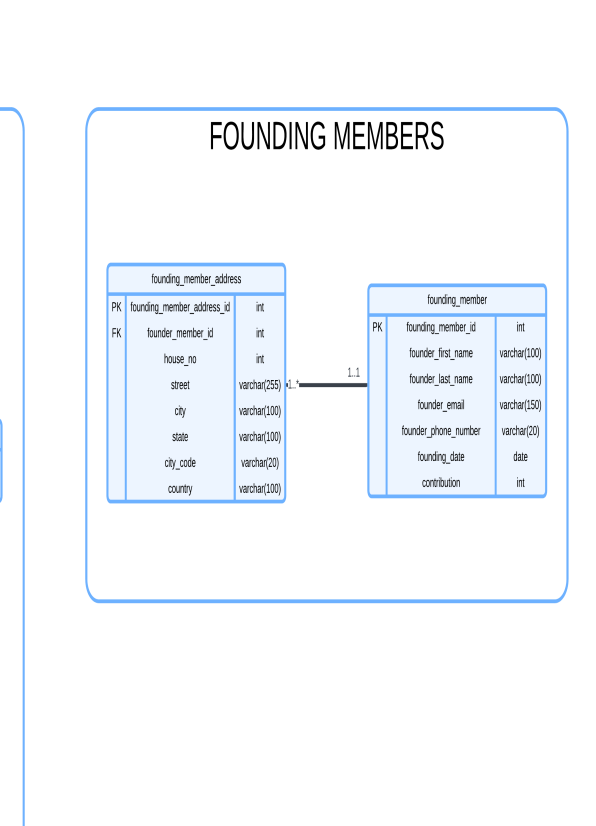
* + It is in 2NF.
  + No non-key attribute is dependent on another non-primary key attribute (eliminating transitive dependency). (Chris, 2022)

*In the current EERD*:

Most tables comply with 3NF, except:

* + **Attendance\_Log**: Attributes like name and email may depend on member\_id. However, these attributes allow tracking of non-member attendees who do not exist in the CRM.
  + **Attendance\_log**: Atrributes like space may depebnd on event\_id, but it is do track of records that show members/non members accessing a space in general
  + **Address Tables**: Separate tables for Country, State, and City could improve normalization. However, given the limited geographic scope of the client's operations, this approach balances practicality with design principes.



## **2.3 Database Schema**

|  |  |  |
| --- | --- | --- |
| **Member** | | |
| **Attribute Name** | **Type** | **Description** |
| Member\_id | int | Primary key |
| First\_name | Varchar (100) | Stores member’s name |
| Last\_name | Varchar(100) | Stores member’s surname |
| Email | Varchar(150) | Stores member’s email |
| Phone\_number | Varchar(20) | Stores member’s phone number |
| Membership\_id | int | Foreign key from **membership table** |
| password | Varchar(255) | Stores member’s password |
| Joining\_date | Date | Stores the date the member joined |
| Subscribed\_newsletter | bool | Stores Yes/No value |

|  |  |  |
| --- | --- | --- |
| **Member\_address** | | |
| **Attribute Name** | **Type** | **Description** |
| Address\_id | int | Primary key |
| Member\_id | int | Foreign key from **member table** |
| House\_no | int | Stores the house number |
| street | Varchar(255) | Stores the street |
| City | Varchar(100) | Stores the city |
| State | Varchar(100) | Stores the state |
| City\_code | Varchar(20) | Stores the city code |
| Country | Varchar(100) | Stores the country |

|  |  |  |
| --- | --- | --- |
| **Membership** | | |
| **Attribute Name** | **Type** | **Description** |
| Membership\_id | int | Primary key |
| Membership\_type | Varchar(50) | Stores membership type |
| Membership\_price | int | Stores the membership price |

|  |  |  |
| --- | --- | --- |
| **Payment** | | |
| **Attribute Name** | **Type** | **Description** |
| Payment\_id | int | Primary key |
| Payment\_reason\_id | int | Foreign key from **payment\_reason table** |
| Payment\_method | char(20) | Stores the payment method |
| Payment\_amount | int | Stores the amount paid |
| Payment\_date | Date | Stores the payment date |

|  |  |  |
| --- | --- | --- |
| **Event** | | |
| **Attribute Name** | **Type** | **Description** |
| Event\_id | Int | Primary key |
| Event\_name | Varchar(255) | Stores the event name |
| Event\_start\_time | Time | Stores the event start time |
| Event\_end\_time | Time | Stores the event end time |
| Event\_date | Date | Stores the event date |
| Event\_type\_id | Int | Foreign key from **event type table** |
| Space\_id | Int | Foreign key from **space table** |
| Expected\_attendance | Int | Expected attendance for an event |
| Space\_hire\_type | Varchar(100) | Type of hire (Partial/Full) |
| Public\_price | Int | Price for non-members |
| Member\_price | Int | Price for Members |

|  |  |  |
| --- | --- | --- |
| **Log\_in** | | |
| **Attribute Name** | **Type** | **Description** |
| Log\_in\_id | Int | Primary key |
| Member\_id | Int | Foreign key from **member table** |
| Log\_in\_time | Time | Stores the login time |
| Log\_in\_date | Date | Date of log-in |
| Log\_in\_device | Varchar(255) | Stores the login device |

|  |  |  |
| --- | --- | --- |
| **Attendance\_log** | | |
| **Attribute Name** | **Type** | **Description** |
| Attendance\_id | Int | Primary key |
| Member\_id | Int | Foreign key from **member table** |
| Attendee\_name | Varchar(255) | Stores the attendee’s name |
| Email | Varchar(255) | Store's the attendee’s email |
| Event\_id | Int | Foreign key from **event table** |
| Space\_id | Int | Foreign key from **space table** |
| Check\_in\_date | Date | Stores the check\_in date |
| Check\_in\_time | Time | Stores the check\_in time |

|  |  |  |
| --- | --- | --- |
| **Inquiries** | | |
| **Attribute Name** | **Type** | **Description** |
| Inquiry\_id | Int | Primary key |
| Member\_id | Int | Foreign key from **member table** |
| Inquiry\_description | Varchar(500) | Stores description of the inquiry |
| Inquiry\_type\_id | Int | Foreign key from **inquiry\_type table** |
| user\_type\_id | Int | Foreign key from **inquiry\_user\_type table** |
| Inquiry\_date | Date | Stores the inquiry date |
| Follow\_up\_count | Int | Number of Follow-ups done from staff to the members |
| status | varchar (10) | Store’s the inquiries’ status |

|  |  |  |
| --- | --- | --- |
| **Inquiry\_user\_type** | | |
| **Attribute Name** | **Type** | **Description** |
| user\_type\_id | Int | Primary key |
| user\_type\_description | Varchar(255) | Stores the inquiry type description |

|  |  |  |
| --- | --- | --- |
| **Inquiry\_type** | | |
| **Attribute Name** | **Type** | **Description** |
| Inquiry\_type\_id | Int | Primary key |
| Inquiry\_type\_description | Varchar(255) | Stores the inquiry type description |

|  |  |  |
| --- | --- | --- |
| **Interests** | | |
| **Attribute Name** | **Type** | **Description** |
| Interest\_id | Int | Primary key |
| Interest\_name | Varchar(100) | Stores the interest depertment name(Sharing, Helping etc.) |

|  |  |  |
| --- | --- | --- |
| **Feedback** | | |
| **Attribute Name** | **Type** | **Description** |
| Feedback\_id | Int | Primary key |
| Member\_id | Int | Foreign key from **member table** |
| Feedback\_description | Varchar(255) | Stores feedback description |

|  |  |  |
| --- | --- | --- |
| **Space** | | |
| **Attribute Name** | **Type** | **Description** |
| Space\_id | Int | Primary key |
| Space\_name | Varchar(255) | Stores the space name |
| Space\_capacity | Int | Stores the space capacity |

|  |  |  |
| --- | --- | --- |
| **Tags** | | |
| **Attribute Name** | **Type** | **Description** |
| Tag\_id | Int | Primary key |
| Tag\_name | Varchar(100) | Stores the tag name(ex: Sports, Science, Health etc) |

|  |  |  |
| --- | --- | --- |
| **Member\_tag** | | |
| **Attribute Name** | **Type** | **Description** |
| Member\_id | Int | Primary key |
| Tag\_id | Int | Foreign key from **tag table** |

|  |  |  |
| --- | --- | --- |
| **Member\_interest** | | |
| **Attribute Name** | **Type** | **Description** |
| Interest\_id | Int | Foreign key from **interest table** |
| Member\_id | Int | Foreign key from **member table** |

|  |  |  |
| --- | --- | --- |
| **Event\_type** | | |
| **Attribute Name** | **Type** | **Description** |
| Event\_type\_id | Int | Primary key |
| Event\_type\_description | Varchar(255) | Stores event type description |

|  |  |  |
| --- | --- | --- |
| **Bookings** | | |
| **Attribute Name** | **Type** | **Description** |
| Bookings\_id | Int | Primary key |
| Payment\_id | Int | Foreign key from **payment table** |
| Member\_id | Int | Foreign key from **member table** |
| Event\_id | Int | Foreign key from **event table** |

|  |  |  |
| --- | --- | --- |
| **Subscriptions** | | |
| **Attribute Name** | **Type** | **Description** |
| Subscription\_id | Int | Primary key |
| Membership\_id | Int | Foreign key from **membership table** |
| Member\_id | Int | Foreign key from **member table** |
| Payment\_id | Int | Foreign key from **payment table** |
| Subscription\_start\_date | Date | Stores the subscription start date |
| Subscription\_end\_date | Date | Stores the subscription end date |

|  |  |  |
| --- | --- | --- |
| **Payment\_reason** | | |
| **Attribute Name** | **Type** | **Description** |
| Payment\_reason\_id | Int | Primary key |
| Payment\_reason\_description | Varchar(255) | Stores the payment reason description |

|  |  |  |
| --- | --- | --- |
| **Founding\_member** | | |
| **Attribute Name** | **Type** | **Description** |
| Founding\_member\_id | Int | Primary key |
| Founder\_first\_name | Varchar(100) | Stores the founding member’s name |
| Founder\_last\_name | Varchar(100) | Stores the founding member’s surname |
| Founder\_email | Varchar(150) | Stores the founding member’s email |
| Founder\_phone\_number | Varchar(20) | Stores the founding member’s phone number |
| Founding\_date | Date | Stores the founding date |
| contribution | Varchar(100) | Stores what the founding member has contributed |

|  |  |  |
| --- | --- | --- |
| **Founding\_member\_address** | | |
| **Attribute Name** | **Type** | **Description** |
| Founding\_member\_address\_id | Int | Primary key |
| Founder\_member\_id | Int | Foreign key from **founder member table** |
| House\_no | Int | Stores the founding member’s house number |
| Street | Varchar(255) | Stores the street |
| City | Varchar(100) | Stores the city |
| State | Varchar(100) | Stores the state |
| City\_code | Varchar(20) | Stores the city code |
| Country | Varchar(100) | Stores the country |

|  |  |  |
| --- | --- | --- |
| **Staff** | | |
| **Attribute Name** | **Type** | **Description** |
| Staff\_id | Int | Primary key |
| Staff\_first\_name | Varchar(100) | Stores the staff’s name |
| Staff\_last\_name | Varchar(100) | Stores the staff’s surname |
| Staff\_email | Varchar(100) | Stores the staff’s email |
| Role\_id | Int | Foreign key from **role table** |
| Workplace\_id | Int | Foreign key from **workplace table** |
| Staff\_contract\_id | Int | Foreign key from **staff contract table** |
| Staff\_clearance\_id | Int | Foreign key from **staff clearance table** |
| salary | Int | Stores the staff’s salary |
| Staff\_salary\_type\_id | Int | Stores the staff’s salary type |
| Notice\_period | Varchar(50) | Stores the notice period |
| Date\_joined | Date | Stores the date joined |

|  |  |  |
| --- | --- | --- |
| **Staff\_attendace** | | |
| **Attribute Name** | **Type** | **Description** |
| Staff\_attendace\_id | Int | Primary key |
| Staff\_id | Int | Foreign key from **staff table** |
| Staff\_check\_in\_time | time | Stores the staff check in time |
| Staff\_check\_out\_time | time | Stores the staff check out time |
| Word\_date | date | Stores the date |

|  |  |  |
| --- | --- | --- |
| **Staff\_address** | | |
| **Attribute Name** | **Type** | **Description** |
| Staff\_address\_id | Int | Primary key |
| Staff\_id | Int | Foreign key from **staff table** |
| House\_number | Int | Stores the house number |
| street | Varchar(255) | Stores the street |
| city | Varchar(100) | Stores the city |
| state | Varchar(255) | Stores the state |
| City\_code | Varchar(20) | Stores the city code |
| country | Varchar(100) | Stores the country |

|  |  |  |
| --- | --- | --- |
| **Role** | | |
| **Attribute Name** | **Type** | **Description** |
| Role\_id | Int | Primary key |
| Rolee\_description | Varchar(255) | Stores the workplace description |

|  |  |  |
| --- | --- | --- |
| **Workplace** | | |
| **Attribute Name** | **Type** | **Description** |
| Workplace\_id | int | Primary key |
| Workplace\_description | Varchar(255) | Stores the workplace description |

|  |  |  |
| --- | --- | --- |
| **Staff\_salary\_type** | | |
| **Attribute Name** | **Type** | **Description** |
| Staff\_salary\_type\_id | int | Primary key |
| Salary\_type\_description | Varchar(255) | Stores the salary type description |

|  |  |  |
| --- | --- | --- |
| **Staff\_clearance** | | |
| **Attribute Name** | **Type** | **Description** |
| Staff\_clearance\_id | int | Primary key |
| Clearance\_description | Varchar(255) | Stores the clearance description |

|  |  |  |
| --- | --- | --- |
| **Staff\_contract** | | |
| **Attribute Name** | **Type** | **Description** |
| Staff\_contract\_id | int | Primary key |
| Contract\_description | Varchar(255) | Stores the contract description |

|  |  |  |
| --- | --- | --- |
| **Staff\_salary\_type** | | |
| **Attribute Name** | **Type** | **Description** |
| Staff\_salary\_type\_id | int | Primary key |
| Salary\_type\_description | Varchar(255) | Stores the salary type |

# **Mapping**

## 3.1 General Usage Patterns

|  |  |  |
| --- | --- | --- |
| **Questions** | **Map to** | **Additional Comments** |
| 1. What are the overall patterns of members’ use of the space over time? | Attendance\_log, Event, Member, Space id | Aggregate attendance\_log by time (hour, day, week, month) |
| 2. Which days of the week see the highest member attendance? | Attendance\_log | Group by weekday  (day of week (check\_in\_time)) and count attendance |
| 3. What times of day are the busiest? | Attendence\_log | Group by the time of day  ( Hour(check\_in\_time) ) and count attendance |
| 4. How do usage patterns vary between weekdays and weekends? | Attendence\_log | Use a CASE statement to categorize days as weekdays or weekends, then compare attendance counts. |
| 5. Are there any seasonal variations in space usage? | Attendence\_log | Aggregate by month or quarter ( month(check\_in\_time) or quarter(check\_in\_time) ) to track seasonal trends. |

## 3.2 Individual Member Usage

|  |  |  |
| --- | --- | --- |
| **Questions** | **Map to** | **Additional Comments** |
| 1) When are individual members visiting the space most frequently? | Attendance\_log, Member | Group by member and time to find frequent visit times. |
| 2 ) Can we track if a particular member's attendance has changed over time? | Attendance\_log, Member | Compare attendance frequency between periods for each member |
| 3 ) Has a member's usage increased significantly, indicating higher engagement? | Attendance\_log, Login, Member | Compare the number of sign\_ins or check\_ins over time for each member |
| 4 ) Has a member been absent for an extended period, suggesting possible attrition risk? | Attendance\_log, Member | Identify members with no check\_ins for a specified time period |
| 5) Can we generate alerts for unusual changes in individual member attendance patterns? | Attendance\_log, Member | Usage of queries to find unusual attendance drops or increases |

## 3.3 Event Participation and Interests

|  |  |  |
| --- | --- | --- |
| **Questions** | **Map to** | **Additional Comments** |
| 1. What types of events are members most interested in (e.g., wellbeing, citizenship, making activities)? | Event, Member\_tags, tags | Aggregate events by type and correlate with member interests. |
| 2. Can we use a system of tags to track areas of interest for members? | Member\_tag, Tags | Join Member\_Tag and Tags to identify areas of member interest based on tags. |
| 3. How many events has each member attended within a specific period? | Attendace\_log, Event | Count distinct events attended by each member in a specific period. |
| 4. Which events have the highest attendance rates? | Attendace\_log, Event | Group by event\_id and count attendance to find the most popular events. |
| 5. Can we identify trends in event participation among different member segments? | Attendace\_log, Event, Member, Member\_interest | Group members by interest segments, then track event participation trends for each segment. |
| 6. How do event interests correlate with members' overall engagement and space usage? | Event, Member\_interest, Attendance\_log | Correlate participation in events with engagement data like number of sign-ins and attendance. |

## 3.4 Audience Segmentation and Personalization

|  |  |  |
| --- | --- | --- |
| **Questions** | **Map to** | **Additional Comments** |
| 1. Can we segment potential members based on their interests and interactions before joining? | Inquiry, member\_tags, tags | Segment members based on their interests(tags) and initial inquiries using tags and interest tables |
| 1. How effective are our current acquisition strategies based on engagement data? | Inquiries | Analyze conversion rates from inquiry to membership, and correlate with engagement levels |
| 1. What is the average volume of digital engagements before converting to membership? | Inquiries | Aggregate Log\_in date for members prior to their joining to calculate digital engagement volume. |
| 1. Can we track and respond to potential members' interests more effectively? | Inquiries, member\_tag, Tags | Track the specific interests or tags associated with potential members through inquiries |
| 1. How can we personalize communication with members based on their segmented interests? | Member\_tag, Tags, | Use member tags and interests to create personalized communication stratagies |

## 3.5 Space Utilisation and Capacity Planning

|  |  |  |
| --- | --- | --- |
| Questions | Map to | Additional Comments |
| 1. What is the average capacity utilisation of the space at different times? | Event, Attendance\_log, Spaces | Calculate average attendance per space over time, then compare with space capacity. |
| 2. How can we optimise workspace allocation to ensure a dynamic and integrated community? | Space | Analyze space usage and optimize allocation based on space capacity and attendance patterns. |
| 3. Are there specific areas within the space that are underutilised? | Attendance\_log, Event, Spaces | Identify underutilized spaces by comparing space attendance rates to their capacity. |
| 4. How can we avoid having segregated areas and promote mixing of different disciplines? | Space, Event | Promote cross-discipline events by analyzing space allocation based on event types and interests. |
| 5. How can we back up our capacity estimates with data for future planning? | Event, Attendance\_log,Space | Use historical attendance and space data to predict future space needs. |

## 3.6 Reporting and Data Integration

|  |  |  |
| --- | --- | --- |
| Question | Map to; | Query Approach |
| 1) Can we generate detailed reports on student member activity for organisational members like ARU? | Member, Attendance\_log, Events | Generate reports by joining Member,Event, and Attendance\_log for student members |
| 2) Can we automate data entry processes and report generation to reduce manual efforts? | All tables | Integrating specialized queries that would generate reports about selected areas of member info, triggered through external queues like the CRM app |
| 3) How easily can we pull reports from the CRM that provide the necessary insights? | All tables | Using special connections from CRM that trigger different events in the database that carries out queries required for report generation |
| 4) Are there any data integrity issues we need to address during integration? | All tables | Handle missing data fields by validating input, respect foreign key constraints to prevent orphaned records, and be cautious when deleting data to avoid unintentional loss or inconsistencies across related tables |

## 3.7 User journey and Acquisition

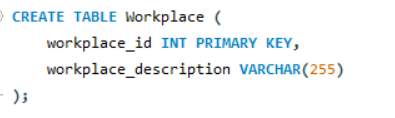
|  |  |  |
| --- | --- | --- |
| Questions | Map to; | Additional Comments |
| 1)  How can we better track potential members' interactions before they join? | Inquiries and attendance\_log | Track records which submit inquiries and attend events/access member space where member id is missing, indicating that the person is not a member, but is a potential client |
| 2)   What tools can help us gather useful data on digital engagements? | Not directly applicable | Could be used by doing some adjustments to the following:  Attendance log  Inquiries  Login  Event  Adding functionalities to track inquiries/events etc. On the online spectrum |
| 3)   What additional insights can tools like Mouseflow and Hotjar provide to improve user  engagement? | Not applicable | These are external analytics tools that track user behavior on websites(not directly related to the database columns) |

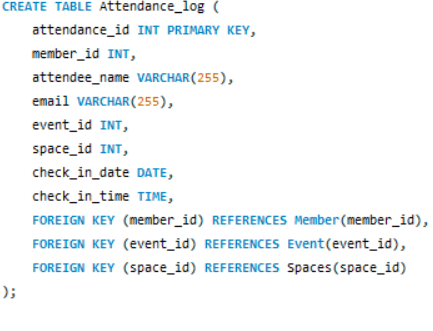
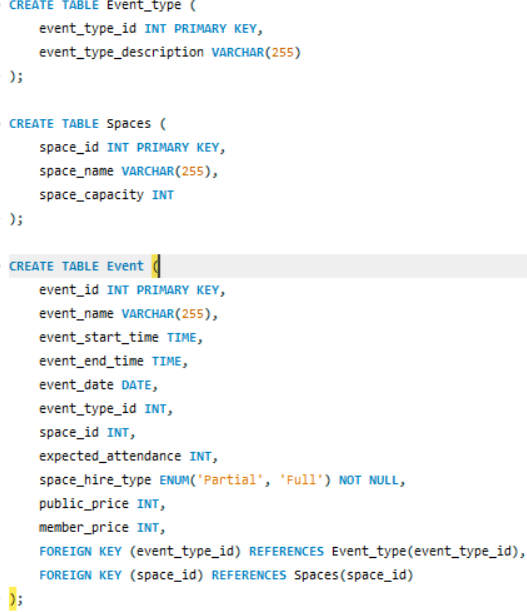
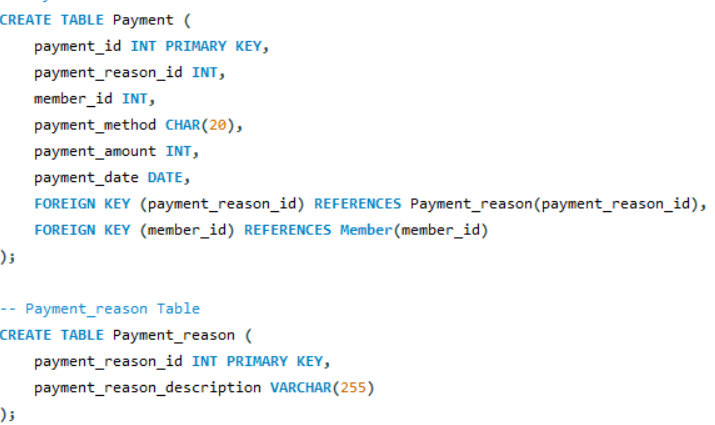
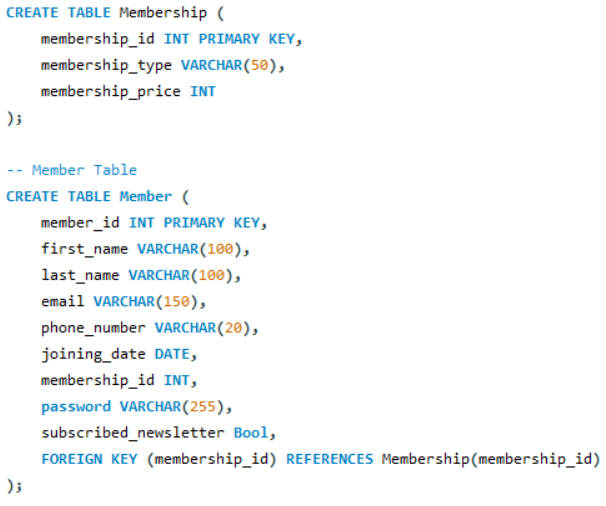
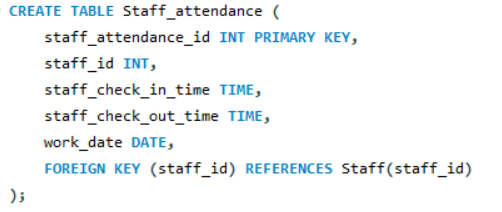
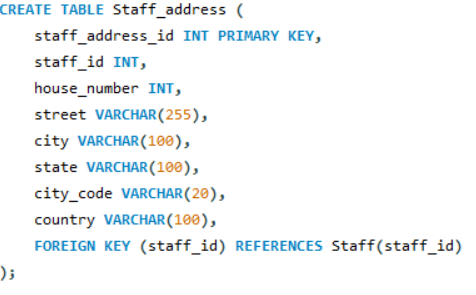
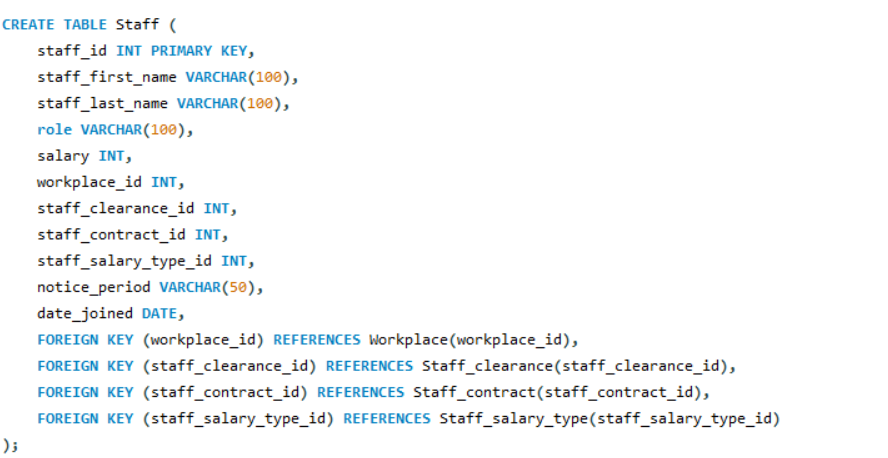
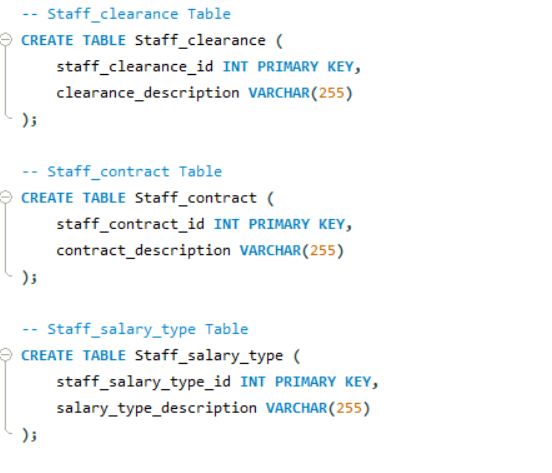
**4.    Database implementation**

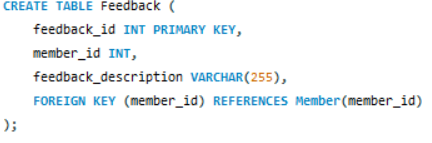
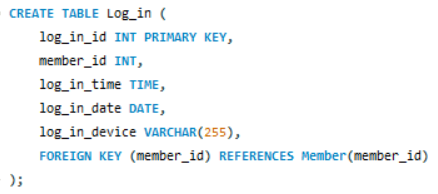
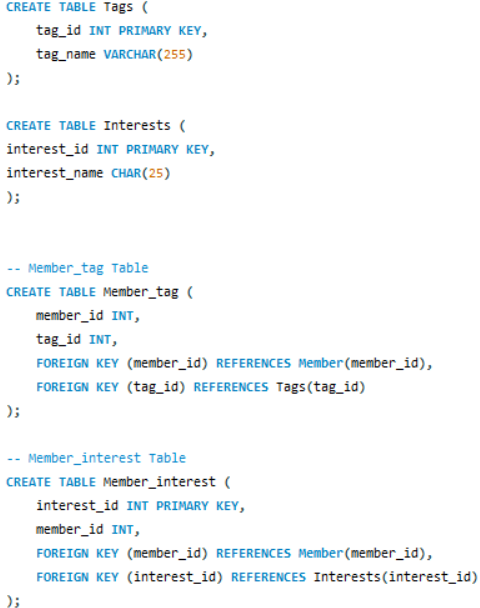
CREATE STATEMENTS:

The process of Create tables taken here is simple:   
First creating the tables with no Foreign Keys then creating the rest of the tables

Staff Members:







Founding Members:



 Tables with Data:

Data in the following tables are sample values and are largely not related to Together Culture Company’s Demographic. Tables are shown in Alphabetical order.

1. Attendance\_log

A screenshot of a computer

Description automatically generated

1. Bookings:

A screenshot of a computer

Description automatically generated

1. Events:

A screenshot of a computer

Description automatically generated

1. Event-Type:

A screenshot of a web page

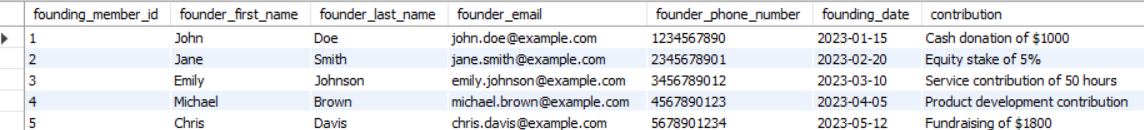
Description automatically generated

1. Feedback:

A screenshot of a computer

Description automatically generated

1. Founding Members:



1. Inquiries:

A screenshot of a computer

Description automatically generated

1. Founding Member Address:

A screenshot of a computer

Description automatically generated

1. Inquiry Type:

A screenshot of a computer

Description automatically generated

1. Interest:

A screenshot of a computer

Description automatically generated

1. Log\_ins:

A screenshot of a computer

Description automatically generated

1. Members:

A screenshot of a computer

Description automatically generated

1. Member\_Interest:

A screenshot of a spreadsheet

Description automatically generated

1. Member\_addresses:

A screenshot of a computer

Description automatically generated

1. Member\_Tags:

A screenshot of a table

Description automatically generated

1. Memberships:

A screenshot of a computer

Description automatically generated

1. Payments:

A screenshot of a computer

Description automatically generated

1. Payment\_reason:

A screenshot of a phone

Description automatically generated

1. Spaces:

A screenshot of a computer

Description automatically generated

1. Staff:

A screenshot of a computer

Description automatically generated

1. Staff\_Address:

A screenshot of a computer

Description automatically generated

1. Staff\_Attendance:

A screenshot of a computer

Description automatically generated

1. Staff\_clearance:

A screenshot of a computer

Description automatically generated

1. Staff\_Contract:

A screenshot of a computer

Description automatically generated

1. Staff Salary Type:

A screenshot of a computer

Description automatically generated

1. Subscriptions:

A screenshot of a computer

Description automatically generated

1. Tags:

A screenshot of a computer

Description automatically generated

1. Workplace:

A screenshot of a computer

Description automatically generated

# **SQL Queries**

(THE FOLLOWING OUTPUT SCREENSHOTS ARE BASED ON THE SAMPLE VALUES IN MY DATABASE)

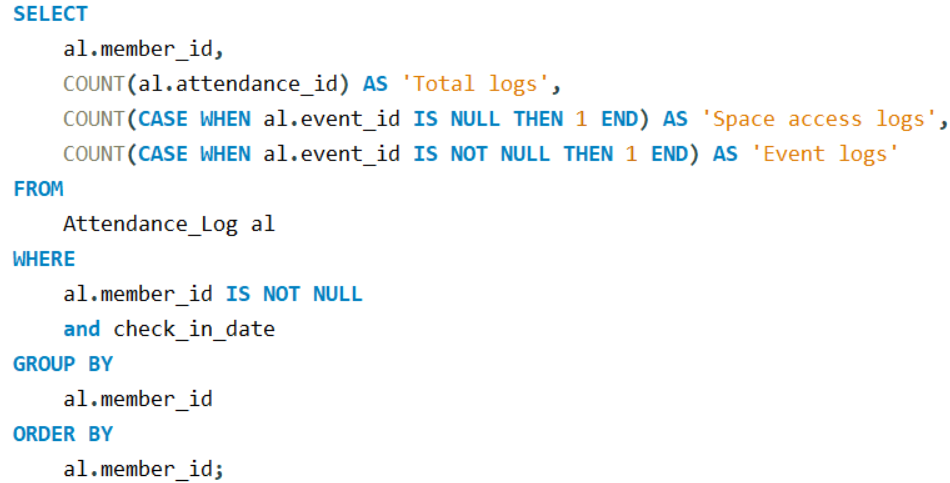
## 5.1 Query 1

### **5.1.2** **Query in natural language (Question from the list provided on Canvas)**

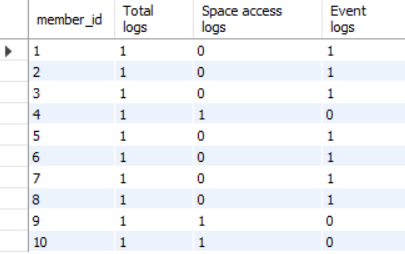
What are the overall patterns of members’ use of the space over time?

### **5.1.3** **SQL Code and output**

Code:



Output:



### **5.1.4** **Explain the output of the data (was this what was predicted?)**

The output displays for each member, their attendance logs of, when they attended an event (Event Logs), or when they were just accessing a space (Space Access). At the Where clause, one could set the checki\_in\_date for a specific date, or a range of dates using BETWEEN to check the above info according to suitable preferences.

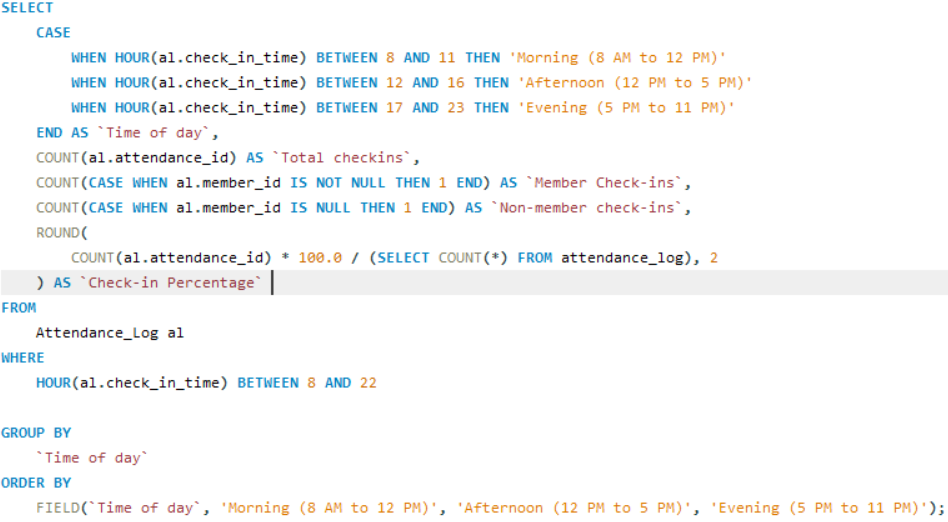
## 5.2 Query 2

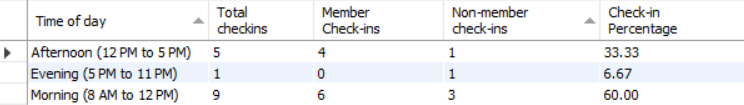
### **5.2.2** **Query in natural language (Question from the list provided on Canvas)**

What times of day are the busiest?

### **5.2.3** **SQL Code and output**

Code:



Output:   


### **5.2.4** **Explain the output of the data (was this what was predicted?)**

Output shows the number of check-ins for a particular time frame of a day, and what type of check in they were (Member or Non-Member) along with Check-In Percentage which gives a concrete ratio of check-ins for those time frames.

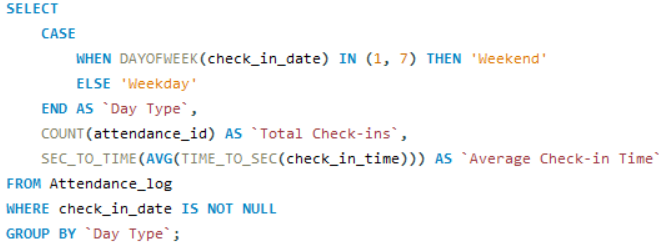
## 5.3 Query 3

### **5.3.2** **Query in natural language (Question from the list provided on Canvas)**

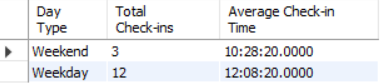
How do usage patterns vary between weekdays and weekends?

### **5.3.3** **SQL Code and output**

Code:



Output:



### **5.3.4** **Explain the output of the data (was this what was predicted?)**

Here for Day Types Weekday or Weekends, it shows how many Check-ins occurred in them, along with the Average check-in Time which would help the admins decide appropriate times to host future events.

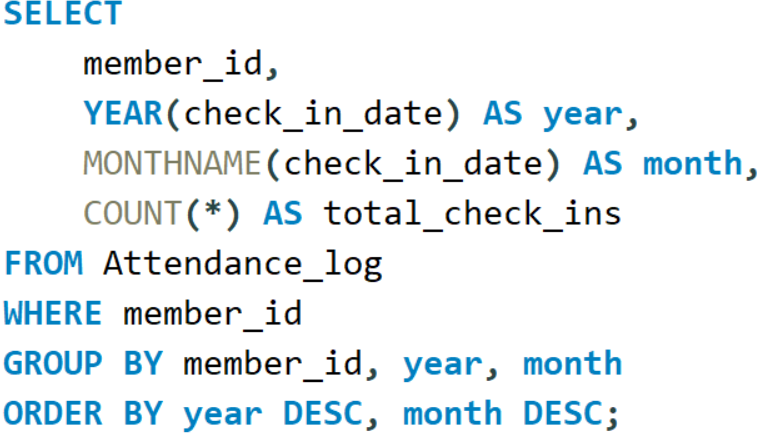
## 5.4 Query 4

### **5.4.2** **Query in natural language (Question from the list provided on Canvas)**

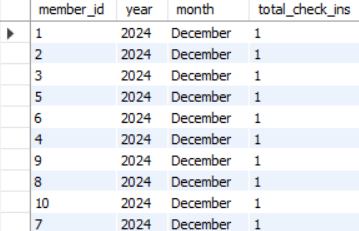
When are individual members visiting the space most frequently?

### **5.4.3** **SQL Code and output**

Code:



Output:



### **5.4.4** **Explain the output of the data (was this what was predicted?)**

The Output here shows each members number of check ins for each month and year they check in, because of the limited sample data in my tables, the output is not much definitive, This would help to understand, on a monthly basis, how many times the member came to the access the space

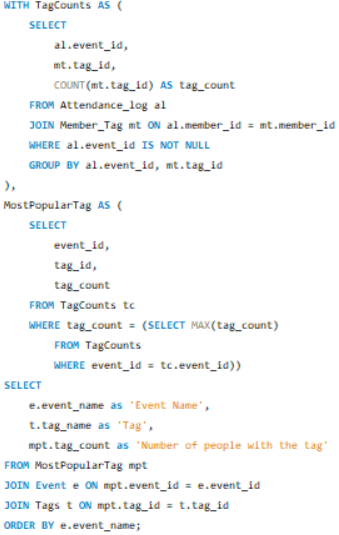
## 5.5 Query 5

### **5.5.2** **Query in natural language (Question from the list provided on Canvas)**

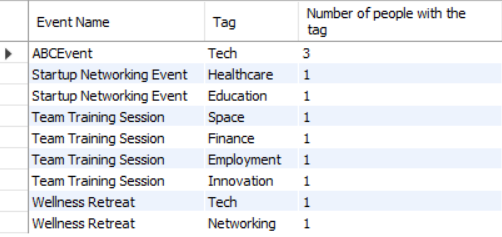
Can we use a system of tags to track areas of interest for members?

### **5.5.3** **SQL Code and output**

Code:



Output:



### **5.5.4** **Explain the output of the data (was this what was predicted?)**

This Output shows for an event, what Tags area of people, and how many members per tag were attracted by an event. This helps the company to suggest appropriate events to members based on the tags they choose and manage the interest of those people accordingly in other areas of business as well.

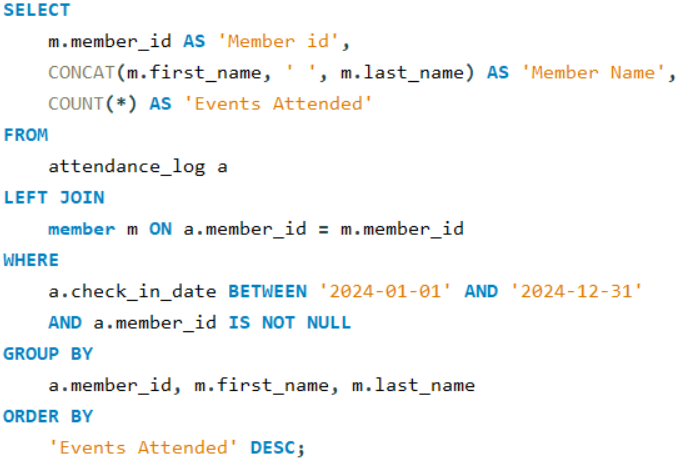
## 5.6 Query 6

### **5.6.2** **Query in natural language (Question from the list provided on Canvas)**

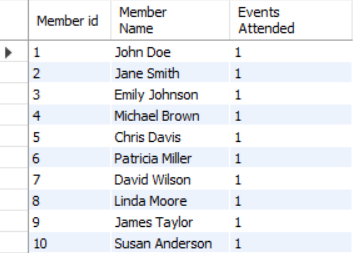
How many events has each member attended within a specific period?

### **5.6.3** **SQL Code and output**

Code:



Output:



### **5.6.4** **Explain the output of the data (was this what was predicted?)**

Here, the output shows for a particular time period which is defined in the Where clause of the SQL query (here from 2024-01-01 to 2024-12-31) which can be adjusted by changing the set dates, for each member, along with their name, how many events they attended.

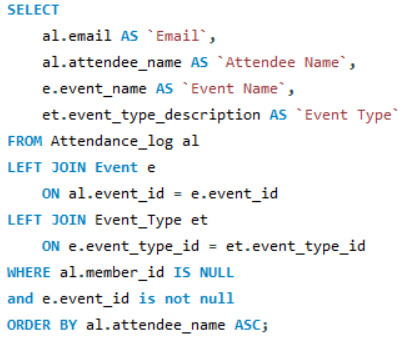
## 5.7 Query 7

### **5.7.2** **Query in natural language (Question from the list provided on Canvas)**

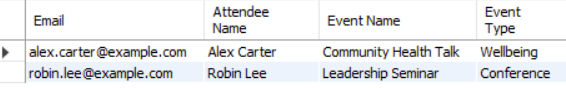
Can we track and respond to potential members' interests more effectively?

### **5.7.3** **SQL Code and output**

Code:



Output:



### **5.7.4** **Explain the output of the data (was this what was predicted?)**

Here, using the attendance table, it takes the information of the people who are not a member, show what events they attended and its event type, using this, admins would use contact of these people to offer them suggestions of events similar to what they attended earlier

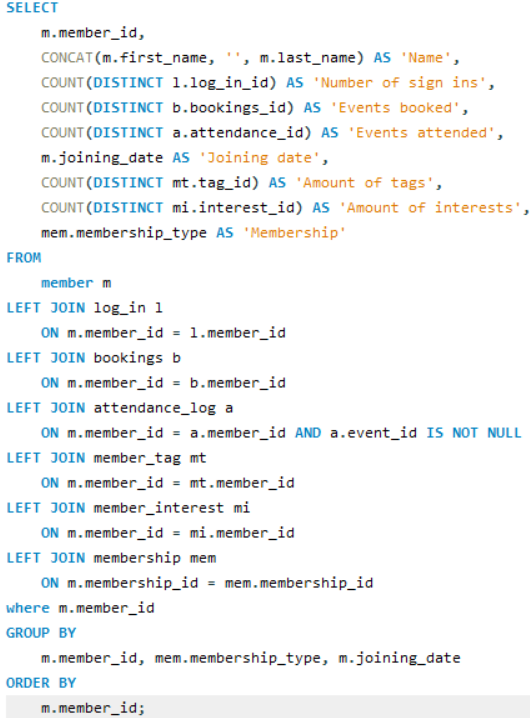
## 5.8 Query 8

### **5.8.2** **Query in natural language (Question from the list provided on Canvas)**

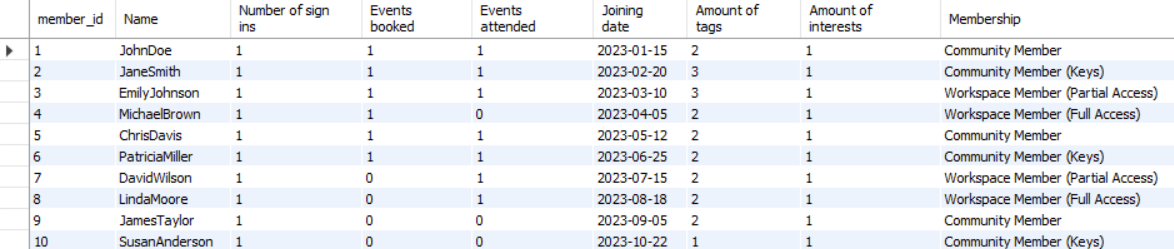
Can we generate detailed reports on student member activity for organizational members like ARU?

**5.8.3**  **SQL Code and output**

Code:



Output:



### **5.8.4** **Explain the output of the data (was this what was predicted?)**

The following output shows for each member, their collected details, like Name, number of digital app sign ins, events attended, booked, membership etc. enabling the company to make reports about a member more easily.

## 5.9 Query 9

### **5.9.1** **For what purpose will this query be used**

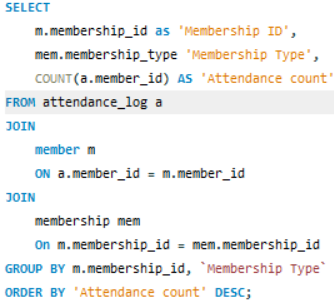
To get a correlation between attendance and membership type

### **5.9.2** **Query in natural language**

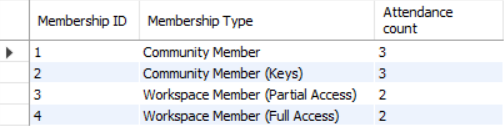
Is there a correlation between a member’s membership type and their attendance.

### **5.9.3** **SQL Code and output**

Code:



Output:



### **5.9.4** **Explain the output of the data (was this what was predicted?)**

This shows Memberships and amount of their relating attendance records. May help the company to analyse if members with specific memberships tend to access the space more or less and updating their memberships perks accordingly.

## 5.10 Query 10

### **5.10.1 For what purpose will this query be used**

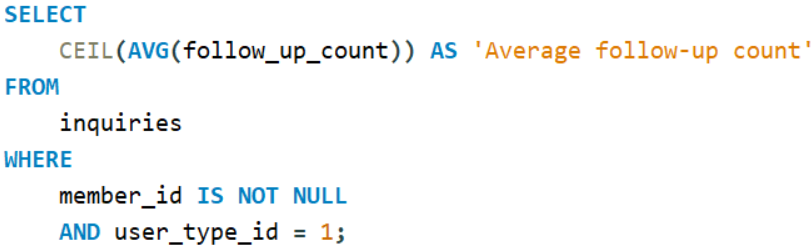
Analyzing the current inquiry strategies

### **5.10.2 Query in natural language**

Tracking how many follow-ups on average it takes to convert a non-member to a member

### **5.10.3 SQL Code and output**

Code:



Output:



### **5.10.4 Explain the output of the data (was this what was predicted?)**

Here the query calculates the average number of follow-ups required to convert a potential member into an active member. It looks at records from the inquiries table where the member id is not null (indicating that the person eventually became a member but the user type is 1 (representing that they were classified as a non-member at the time of the inquiry). This helps in analyzing the transition phase when individuals were still potential members. By calculating the average follow\_up\_count for these records, we gain insights into the effort required to convert non-members into members.

Conclusion

In conclusion, we have successfully met the learning outcomes for the database design and implementation module. This project provided valuable experience in designing an efficient CRM database, analyzing data relationships, and refining the design to meet specific requirements. By applying normalization techniques and creating clear entity relationships, we have strengthened essential skills in database design. Moving forward, the author now critically evaluates existing systems, identifying opportunities for improvement and applying creative solutions to enhance performance and usability. This project has laid a solid foundation for our future work in software and web development.

References

DBMS ER diagram (UML notation) EERD | Lucidchart (2015) Lucid.app. Available at: <https://lucid.app/lucidchart/b8392390-cd99-491c-8d79-2616095fb149/edit?invitationId=inv_968360d9-7587-4168-8179-46b9618d4e16&page=0_0> [Accessed 5 November 2024].

Chris, K. (2022) Database Normalization – Normal Forms 1nf 2nf 3nf Table Examples, freeCodeCamp.org. Available at: <https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/>. [Accessed 11 November 2024].

ER diagrams vs. EER diagrams: What’s the difference? (2018) Nulab. Available at: <https://nulab.com/learn/software-development/er-diagrams-vs-eer-diagrams-whats-the-difference/>. [Accessed 8 November 2024].

CAMBRIDGE, T.C. (2024) Together Culture CAMBRIDGE, Together Culture CAMBRIDGE. Available at: https://www.togetherculture.com. [Accessed 18 October 2024].