Agency 241: Database Management

By: ***Neha Siddiqui, Therese Florido, Taylor Lewis, Micah Desai***

INTRODUCTION

In the current job market, it is more important than ever that students are given an opportunity to build their resumes and portfolios while still in school. With most entry-level job and internship descriptions including the paradoxical requirement of “previous experience,” students should have the means to gain said experience while attending college to prepare them to enter the workforce.

University clubs and student organizations address this need by providing professional development opportunities to their members. Some help students connect with peers within their area of study and industry professionals who are well into their careers. Student organizations also hold professional development workshops and networking events to further supply students with support and guidance.

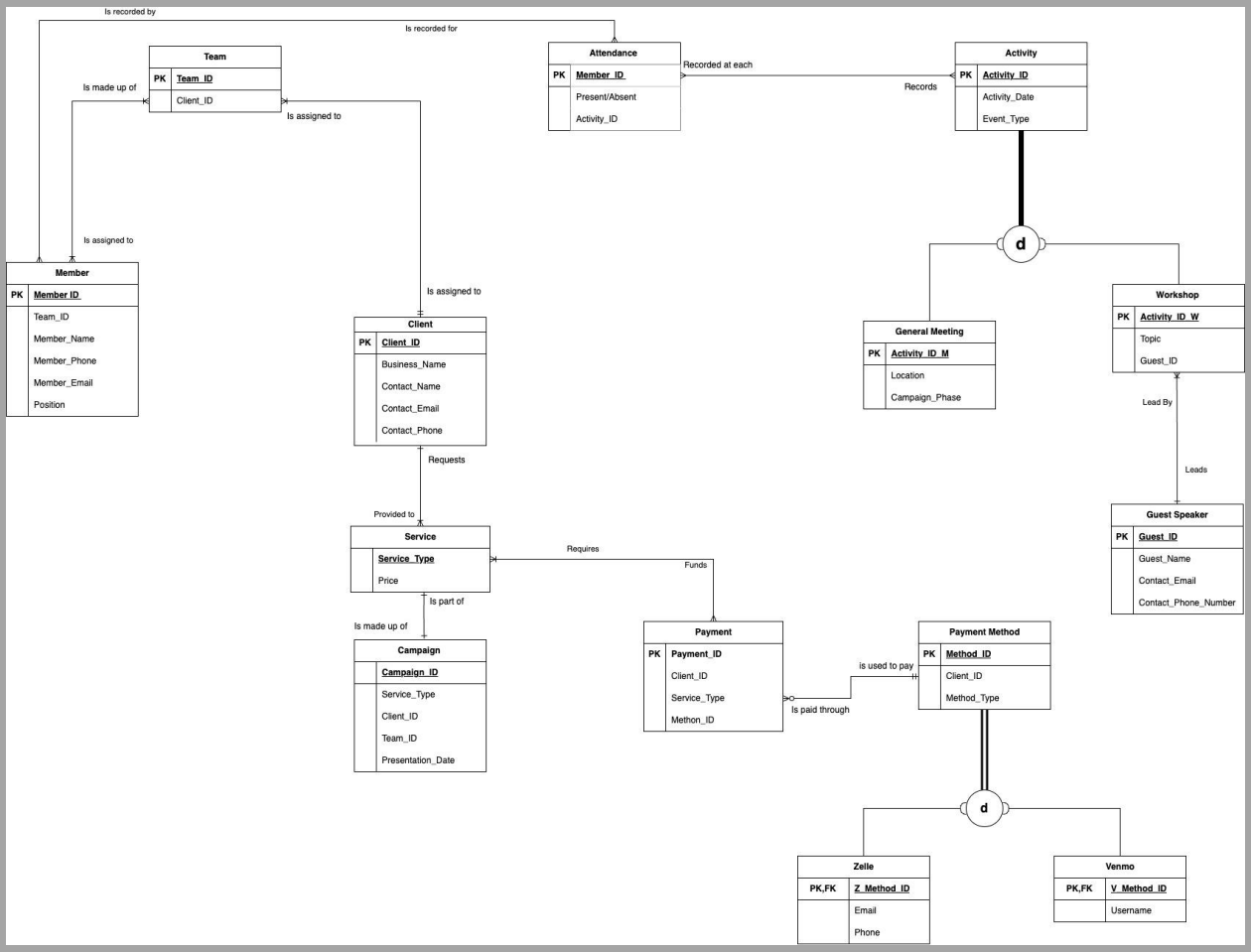
Agency 241 is a student-led marketing agency at CSU Long Beach. The organization aims to provide CSULB students of all majors with **real-world experience working with local businesses**. Founded in 2016 by the American Marketing Association, Long Beach, Agency 241 is well-established and fairly well-known in the Beach community as a way for students to gain professional development, resume-building skills, and internship experience.

Each semester, the student agency reaches out to local businesses to provide marketing services to about **6 business clients**. Student **teams of 7 work on 10-week campaigns** throughout the semester to accomplish a range of marketing activities for an assigned client. Student members are chosen through an interview process to fill certain roles within each team. Campaigns are composed of a **range of services** depending on the client’s needs and the team’s abilities. The client **pays** a certain amount for each service the agency provides. The agency also **holds work and social activities** catered to members, such as weekly general meetings for members to work together in person, social events for community building and engagement, and workshops to train members throughout the semester.

As Agency 241 has grown over the years, the organization has accumulated a large amount of student and client data. Currently, this data is organized in spreadsheets and documents, however, there is no one place to access the accumulated wealth of information. With a current participation of over 40 students and 6 clients each semester, there is a **new need for centralized organization** and access to the information available.

DIAGRAMS

1. Conceptual Model:



In our conceptual model of the Agency 241 database, we included **14 entities** with appropriate relationships.

Members are the core of the program, so all entities stem from their participation. Members are identified by a Member_ID. Each member is assigned to a Team for each program term they participate in. Member name and contact information are taken, and each member is given a position within their Team, such as Account Manager, Strategic Planner, Copywriter, Website Designer, Graphic Designer, Social Media Coordinator, or Photographer.

Teams are made up of about 7 members. Teams are identified by a Team_ID and are assigned one and only one Client to work with.

The **Client** entity represents a business that the agency works with at any given time. Clients are identified by a Client_ID, and the business's name, point of contact, and the contact information for each business's point of contact are recorded.

Each Client requests one or multiple services. **Services** are identified by Service_Type, but the actual service differs for every Client. Services have an associated price depending on the service performed.

A **Campaign** is identified by a Campaign_ID. Campaigns record what service a specific Client requested and the Team that performs the service. Campaigns are reported on and presented at the end of a term, and the date that the team presents is recorded.

Each client's Payment funds one or multiple Services that they requested. A **Payment** is identified by a Payment_ID and records the Client who makes the payment, the Service they are paying for, and the Payment Method they are using. The Client, Service, and Payment Method are recorded for each Payment.

A **Payment Method** is unique to the Client that pays, so the Client is recorded along with the Payment Method Type.

There are two subtypes of Payment Method, or **Payment Method Types: Zelle and Venmo**. Each of these has a different ID number associated with the specific payment made through that Method. A Payment must be made through one of the two methods and only one of the two; Payments cannot be split between them. Clients who pay through Zelle must provide their unique Email and/or Phone. Clients who pay through Venmo must provide their unique Venmo Username.

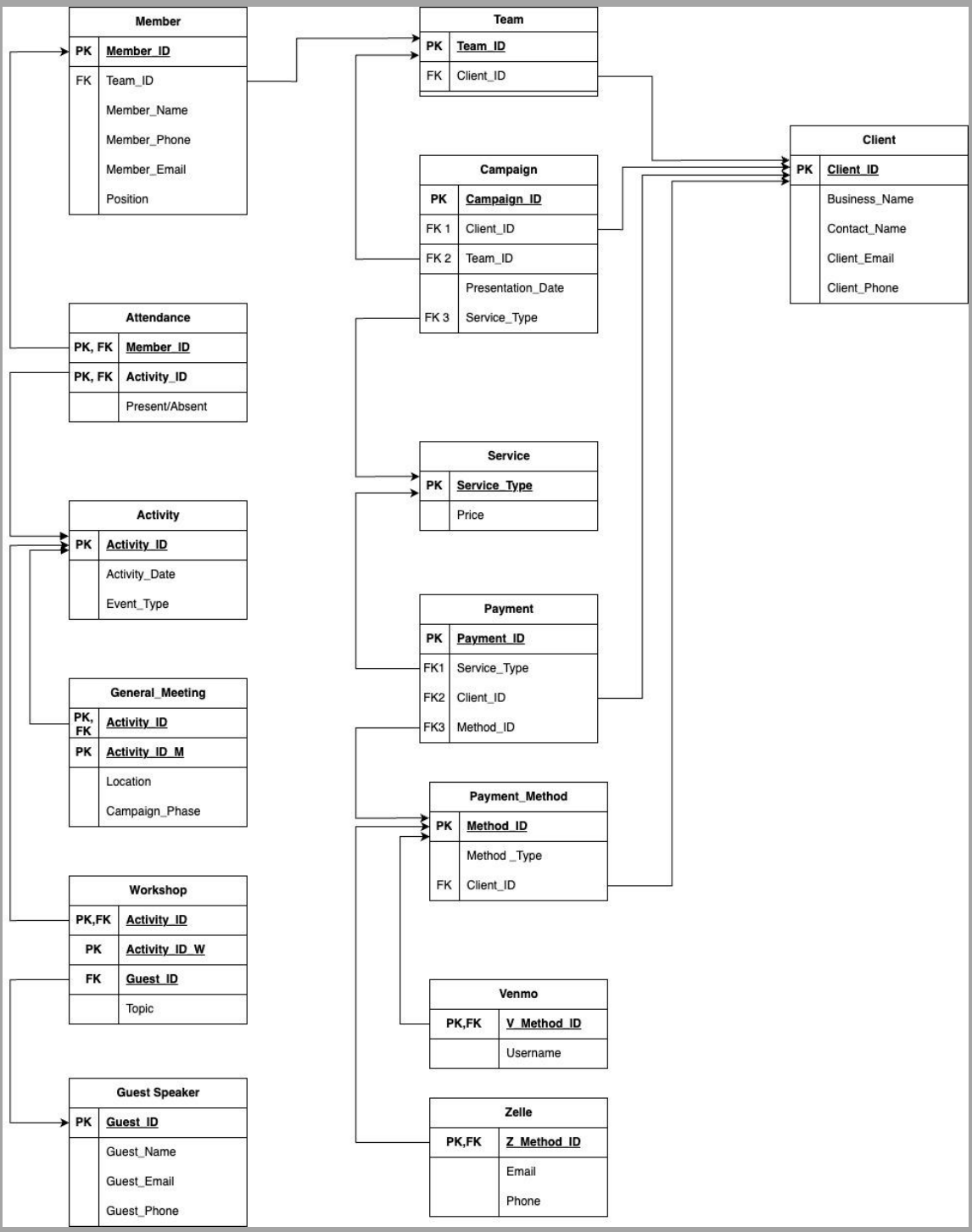
Going back to the Member entity, each Member is recorded as present or absent under **Attendance**. Each instance of Attendance is identified by the Member_ID of the member attending (or not attending). There are events and activities held by the program that members can attend. Any certain member's attendance at any certain Activity is recorded.

An **Activity** is identified by an Activity_ID. The date the Activity occurs on and the type of event being held are recorded. There are two subtypes of Activities: General Meetings and Workshops. Activities can fall under either of these categories, but not both. Some Activities may not be recorded as either of these, such as social events and campaign presentations.

General Meetings are uniquely identified by their Activity_ID_M. The location of the meeting and the phase that groups should be in within their campaigns are recorded (for example, execution and wrap-up phases).

Workshops can be identified by a unique Activity_ID_W. All Workshops are held over Zoom, so location is not recorded. Workshops have topics that are covered by a Guest Speaker. A Guest **Speaker** can lead one or multiple workshops and is identified by a Guest_ID. The Guest's name and contact information are recorded.

2. Logical Model:



In our logical model of the Agency 241 database, we added Primary and Foreign Key indicators next to each attribute as necessary. We then linked the entities from each Foreign Key to the

related entity's Primary Key. As you can see, most of the entities are linked to the Client, Activity, or Payment Method IDs.

All entities exhibit full dependencies. There are no Partial Dependencies because all non-key attributes are dependent on the full primary key. There are no Transitive Dependencies because non-key attributes do not depend on other non-key attributes.

Attendance, General_Meeting, and Workshop have composite primary keys.

- ★ **Attendance Composite Primary Key:** Member_ID, Activity_ID
- ★ **General_Meeting Composite Key:** Activity_ID, Activity_ID_M
- ★ **Workshop Composite Key:** Activity_ID, Activity_ID_W

SUPPORTING DOCUMENTATION

1. Data Dictionary:

Entity: Member
This entity contains members of the organization.

Column	Data Type	Description	Range	Required	PK/FK
Member_ID	INT	ID Number assigned to each member	000000001-999 999999	Y	PK
Team_ID	INT	ID Number to identify the team	000000001-999 999999	Y	FK
Member_Name	VARCHAR	Name of the student who is participating		Y	
Member_Phone	VARCHAR	Member's Phone Number		Y	
Member_Email	VARCHAR	Member's Email Address		Y	
Position	VARCHAR	The role assigned to the Member		Y	

Entity: Team
This entity contains the teams of the organization.

Column	Data Type	Description	Range	Required	PK/FK
Team_ID	INT	ID Number to identify the team	0001-9999	Y	FK
Client_ID	INT	ID Number to identify the client	0001-9999	Y	
Business_Name	VARCHAR	Name of the business client			
Contact_Name	VARCHAR	Name of the business		Y	

		owner or representative who is participating			
Client_Phone	VARCHAR	Client's Phone Number		Y	
Client_Email	VARCHAR	Client's Email Address		Y	

Entity: Client

This entity contains the clients of the organization.

Column	Data Type	Description	Range	Required	PK/FK
Client_ID	INT	ID Number to identify the team	0001-9999	Y	PK
Client_Name	VARCHAR	Name of the client who is participating		Y	
Client_Phone	VARCHAR	Client's Phone Number		Y	
Client_Email	VARCHAR	Client's Email Address		Y	

Entity: Attendance

This entity contains the attendance recorded at each activity.

Column	Data Type	Description	Range	Required	PK/FK
Member _ID	INT	ID Number associated the team	0001-9999	Y	PK
Service_Type	VARCHAR	The type of service that is being provided to client.		Y	
Campaign_ID	INT	ID Number associated to a campaign.		Y	
Price					

Entity: Activity
This entity contains the activities of the organization.

Column	Data Type	Description	Range	Required	PK/FK
Activity _ID	INT	ID Number associated the team	0001-9999	Y	PK
Date	VARCHAR	ID Number associated with a campaign.		Y	
Event_Type	VARCHAR	The type of event hosted		Y	

Entity: General/ Weekly Meeting
This entity contains the general meeting activity.

Column	Data Type	Description	Range	Required	PK/FK
Activity _ID _M	INT	ID Number associated to general meeting	0001-9999	Y	PK
Location	VARCHAR	Location of the Event		N	
Campaign_Phase	VARCHAR	Phase of the Campaign		N	

Entity: Workshop
This entity contains the teams of the organization.

Column	Data Type	Description	Range	Required	PK/FK
Activity _ID _Q	INT	ID Number associated the team	0001-9999	Y	PK
Location	VARCHAR	Meeting Location		N	
Campaign_Phase	VARCHAR	Phase of the campaign		N	

Entity: Workshop
This entity contains the teams of the organization.

Column	Data Type	Description	Range	Required	PK/FK
Activity _ID _W	INT	ID Number	0001-9999	Y	PK

		associated the workshop			
Topic	VARCHAR	Topic of workshop being held		Y	
Guest_ID	INT	ID of Guest Speaker at the Workshop.		Y	

Entity: Guest Speaker
This entity contains the guest speakers invited by the organization

Column	Data Type	Description	Range	Required	PK/FK
Guest_ID	INT	ID Number associated guest.	0001-9999	Y	PK
Guest_Name	VARCHAR	Name of guest speaker		Y	
Contact_ Email	INT	Email Address of guest speaker.		Y	
Contact_Phone_Number	VARCHAR	Phone number of guest speaker.		N	

Entity: Service
This entity contains the service types provided by the organization

Column	Data Type	Description	Range	Required	PK/FK
Service_Type	VARCHAR	Type of service featured in the campaign.		Y	PK
Price	DECIMAL	Fee for the type of service.		Y	

Entity: Campaign
This entity contains the campaigns that the services are made up of

Column	Data Type	Description	Range	Required	PK/FK
Campaign_ID	VARCHAR	Type of service featured in the campaign.		Y	PK
Service_Type	VARCHAR	Type of service featured in the campaign.		Y	
Client_ID	INT	ID Number to identify the team	0001-9999	Y	
Team_ID	INT	ID Number to identify the team	0001-9999	Y	
Presentation_Date	DATE	Date when the campaign is presented.		Y	

Entity: Payment
This entity contains the payment of each service

Column	Data Type	Description	Range	Required	PK/FK
Payment_ID	INT	ID of payment of services	0001-9999	Y	PK
Client_ID	INT	ID Number to identify the team	0001-9999	Y	
Service_Type	VARCHAR	Type of service featured in the campaign.		Y	
Method_ID	INT	ID for method of payment	0001-9999	Y	

Entity: Payment Method
This entity contains the payment method of each service

Column	Data Type	Description	Range	Required	PK/FK
Method_ID	INT	ID for method of payment	0001-9999	Y	PK
Client_ID	INT	ID Number to identify the team	0001-9999	Y	
Method_Type	VARCHAR	Type of payment method.		Y	

Entity: Zelle
This entity contains the Zelle payment method of each service

Column	Data Type	Description	Range	Required	PK/FK
Z_Method_ID	INT	Payment method ID for Zelle payments	0001-9999	Y	PK/ FK
Email	VARCHAR	Email of Zelle transaction.		Y	
Phone	VARCHAR	Phone number of Zelle transaction.		Y	

Entity: Venmo
This entity contains the Venmo payment method of each service

Column	Data Type	Description	Range	Required	PK/FK
V_Method_ID	INT	Payment method ID for Venmo payments	0001-9999	Y	PK/ FK
Username	VARCHAR	Username of Zelle transaction.		Y	

2. **Database Purpose:**

As previously mentioned, Agency 241 has accumulated a high volume of information over the years. As the agency expands, it would be useful to have a database to keep track of information such as what businesses have participated as clients and what services they have requested, what students have participated as members and what their roles were, as well as what events the organization has held in the past. As Agency 241 becomes more and more similar to a real-world marketing agency, it becomes more and more necessary to enhance its existing structure and implement new initiatives to support members and clients.

Some questions this database could answer are:

- ★ What contact information should be used to reach a certain member, client, or guest speaker?
- ★ Where is the next event planned to take place?
- ★ What events are most successful based on member attendance?
- ★ Which members have not attended enough events?
- ★ What services are most requested by clients?
- ★ How much profit is being made each term?

This information is useful in both long-term and short-term cases. A student might need to quickly find the correct email to contact their assigned client on any given day. When planning what events will be held or what services can be provided for the next term, it may be useful to see how much money is available to use in order to decide where to allocate it.

In this case, a successful database would benefit Agency 241 by providing more structure when planning and executing each term. It would enhance the agency's ability to reach clients who need its services, understand what initiatives are most effective and useful for members, and plan where to distribute funds each term.

SQL EXAMPLE QUERIES

To create our Oracle database, we used a series of queries in SQL to create and insert tables and dummy data, which can be found in the 'DATABASE INSERT COMMANDS' section at the end of the report. Using this data, we ran the following queries to complete the tasks related to the questions outlined in the 'Database Purpose' section.

1. **Team_Member_Summary**
    - ★ **Task:** See details about a specific team to know which members are assigned to a specific team and which client that team will be working with.
    - ★ **Output:** Returns members in a specific team, what business they represent, and their positions.

Written Query

```
---
 SELECT
   m.Member_Name,
   m.Position,
   c.Business_Name AS Represented_Business,
   t.Team_ID
FROM
   Member m
JOIN
   Team t ON m.Team_ID = t.Team_ID
JOIN
   Client c ON t.Client_ID = c.Client_ID
WHERE
    M.Team_ID = 2741
ORDER BY M.Position;
```

RESULTS

	MEMBER_NAME	POSITION	REPRESENTED_BUSINESS	TEAM_ID
1	Reese Lee	Account Manager	BrightFuture Marketing	2741
2	Taylor Lee	Copywriter	BrightFuture Marketing	2741
3	Reese Miller	Graphic Designer	BrightFuture Marketing	2741
4	Blake Martinez	Photographer	BrightFuture Marketing	2741
5	Avery Thomas	Social Media Coordinator	BrightFuture Marketing	2741
6	Jordan Lee	Strategic Planner	BrightFuture Marketing	2741
7	Quinn Smith	Website Designer	BrightFuture Marketing	2741

2. **TOP_5_ATTENDED_ACTIVITIES**
    - ★ **Task:** See what activities were the most popular. This can help to determine what activities to continue to hold in the future.
    - ★ **Output:** Returns the top 5 events based on attendance.

Written Query

```
SELECT
   a.Activity_ID,
   a.Event_Type AS Activity_Name,
   a.Activity_Date,
   COUNT(*) AS Attendance_Count
FROM
   Attendance at
JOIN
   Activity a ON at.Activity_ID = a.Activity_ID
WHERE
   at.PresentAbsent = 'Y'
GROUP BY
   a.Activity_ID, a.Event_Type, a.Activity_Date
ORDER BY
   Attendance_Count DESC
FETCH FIRST 5 ROWS ONLY;
```

RESULTS

	ACTIVITY_ID	ACTIVITY_NAME	ACTIVITY_DATE	ATTENDANCE_COUNT
1	248	Workshop	15/01/25	4
2	523	Workshop	12/03/25	4
3	180	Workshop	19/03/25	4
4	405	General Meeting	12/03/25	4
5	427	General Meeting	12/03/25	4

3. **TEAM_LEADERS**
    - ★ **Task:** The Directors need to contact Account Managers to get progress reports and pass along information.
    - ★ **Output:** Returns the account managers of each team and their contact information.

Written Query

```
SELECT
   t.Team_ID,
   m.Member_ID,
   m.Member_Name,
   m.Position,
   m.Member_Email,
   m.Member_Phone
FROM
   Member m
JOIN
   Team t ON m.Team_ID = t.Team_ID
WHERE
   LOWER(m.Position) IN ('account manager');
```

RESULTS

	TEAM_ID	MEMBER_ID	MEMBER_NAME	POSITION	MEMBER_EMAIL	MEMBER_PHONE
1	2741	346138	Reese Lee	Account Manager	reese.lee@example.com	555-123-3578
2	8395	845633	Drew Garcia	Account Manager	drew.garcia@example.com	555-123-2676
3	1427	372053	Sydney Lopez	Account Manager	sydney.lopez@example.com	555-123-8881
4	9063	937816	Peyton Davis	Account Manager	peyton.davis@example.com	555-123-8233
5	3178	190098	Peyton Lee	Account Manager	peyton.lee@example.com	555-123-5948
6	4812	450406	Casey Davis	Account Manager	casey.davis@example.com	555-123-5573

4. **MEMBERS _WITH_ABSENCES**
    - ★ **Task:** Members are allowed 2 absences. We need to see which members have exceeded this number when administering strikes or warnings.
    - ★ **Output:** Lists members who have missed 2 or more events or meetings

Written Query

```
 SELECT
    m.Member_ID,
    m.Member_Name,
    m.Member_Email,
    COUNT(*) AS Absence_Count
FROM
    Attendance a
JOIN
    Member m ON a.Member_ID = m.Member_ID
WHERE
    a.PresentAbsent = 'N'
GROUP BY
    m.Member_ID, m.Member_Name, m.Member_Email
HAVING
    COUNT(*) >= 2;
```

RESULTS

	MEMBER_ID	MEMBER_NAME	MEMBER_EMAIL	ABSENCE_COUNT
1	221725	Jordan Garcia	jordan.garcia@example.com	2
2	845633	Drew Garcia	drew.garcia@example.com	2
3	938429	Quinn Smith	quinn.smith@example.com	2
4	783942	Avery Thomas	avery.thomas@example.com	2
5	829861	Riley Miller	riley.miller@example.com	2

5. **REVENUE_BY_SEMESTER**
    - ★ **Task:** The agency needs to see how much revenue is made in a certain semester to conduct financial planning and budgeting.
    - ★ **Output:** Shows total revenue from all clients.

Written Query

```
SELECT
   EXTRACT(YEAR FROM c.Presentation_Date) AS Revenue_Year,
   CASE
       WHEN EXTRACT(MONTH FROM c.Presentation_Date) BETWEEN 1 AND 6 THEN 'Spring'
       ELSE 'Fall'
   END AS Semester,
   SUM(s.Price) AS Total_Revenue
FROM
   Payment p
JOIN
   Service s ON p.Service_Type = s.Service_Type
JOIN
   Campaign c ON p.Client_ID = c.Client_ID AND p.Service_Type = c.Service_Type
GROUP BY
   EXTRACT(YEAR FROM c.Presentation_Date),
   CASE
       WHEN EXTRACT(MONTH FROM c.Presentation_Date) BETWEEN 1 AND 6 THEN 'Spring'
       ELSE 'Fall'
   END;
```

RESULTS

	REVENUE_YEAR	SEMESTER	TOTAL_REVENUE
1	2025	Spring	2950

6. **WORKSHOP_OVERVIEW_REPORT**
    - ★ **Task:** The Directors need to see how many workshops have been held and how many members attended. This is helpful when creating an end-of-year report on what the agency accomplished and how it helps to support its members.
    - ★ **Output:** Lists workshops by date, guest speakers present, and members that attended.

Written Query

```
SELECT
   a.Activity_ID,
   a.Activity_Date,
   w.Topic AS Workshop_Topic,
   g.Guest_Name AS Guest_Speaker,
   COUNT(DISTINCT at.Member_ID) AS Total_Attendees
FROM
   Workshop w
JOIN
   Activity a ON w.Activity_ID_W = a.Activity_ID
JOIN
   Guest_Speaker g ON w.Guest_ID = g.Guest_ID
JOIN
   Attendance at ON a.Activity_ID = at.Activity_ID
JOIN
   Member m ON at.Member_ID = m.Member_ID
WHERE
   a.Event_Type = 'Workshop'
   AND EXTRACT(YEAR FROM a.Activity_Date) = 2025
   AND at.PresentAbsent = 'Y'
GROUP BY
   a.Activity_ID, a.Activity_Date, w.Topic, g.Guest_Name
ORDER BY
   a.Activity_Date;
```

RESULTS

	ACTIVITY_ID	ACTIVITY_DATE	WORKSHOP_TOPIC	GUEST_SPEAKER	TOTAL_ATTENDEES
1	248	15/01/25	SEO Best Practices	Taylor Brooks	4
2	872	22/01/25	Client Retention Tactics	Jamie Chen	2
3	361	19/02/25	Client Retention Tactics	Riley Nguyen	4
4	523	12/03/25	Marketing Automation	Alex Morgan	4
5	180	19/03/25	Social Media Analytics	Jordan Cruz	4

DATABASE INSERT COMMANDS

OracleDB
User: DataAndTheDynamos
Password:*******

```
-- CLIENT table ---
CREATE TABLE Client (
    Client_ID       VARCHAR2(10) PRIMARY KEY,
    Business_Name   VARCHAR2(100),
    Contact_Name    VARCHAR2(100),
    Client_Email    VARCHAR2(100),
    Client_Phone    VARCHAR2(20));

-- TEAM table ---
CREATE TABLE Team (
    Team_ID   VARCHAR2(10) PRIMARY KEY,
    Client_ID VARCHAR2(10),
    FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID));

-- MEMBER table ---
CREATE TABLE Member (
    Member_ID     VARCHAR2(10) PRIMARY KEY,
    Team_ID       VARCHAR2(10),
    Member_Name   VARCHAR2(100),
    Member_Phone  VARCHAR2(20),
    Member_Email  VARCHAR2(100),
    Position      VARCHAR2(50),
    FOREIGN KEY (Team_ID) REFERENCES Team(Team_ID));

---ATTENDANCE table ---
CREATE TABLE Attendance (
    Member_ID      VARCHAR2(10),
    Activity_ID    VARCHAR2(10),
    PresentAbsent  CHAR(1) CHECK (PresentAbsent IN ('Y', 'N')),
    PRIMARY KEY (Member_ID, Activity_ID),
    FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID),
    FOREIGN KEY (Activity_ID) REFERENCES Activity(Activity_ID));
```

```
-- SERVICE table ---
CREATE TABLE Service (
    Service_Type VARCHAR2(50) PRIMARY KEY,
    Price        NUMBER(10, 2));

-- CAMPAIGN table ---
CREATE TABLE Campaign (
    Campaign_ID      VARCHAR2(10) PRIMARY KEY,
    Client_ID        VARCHAR2(10),
    Team_ID          VARCHAR2(10),
    Service_Type     VARCHAR2(50),
    Presentation_Date DATE,
    FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID),
    FOREIGN KEY (Team_ID) REFERENCES Team(Team_ID),
    FOREIGN KEY (Service_Type) REFERENCES Service(Service_Type));

-- PAYMENT_METHOD table ---
CREATE TABLE Payment_Method (
    Method_ID    VARCHAR2(10) PRIMARY KEY,
    Client_ID    VARCHAR2(10),
    Method_Type  VARCHAR2(50),
    FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID));

-- PAYMENT table ---
CREATE TABLE Payment (
    Payment_ID   VARCHAR2(10) PRIMARY KEY,
    Service_Type VARCHAR2(50),
    Client_ID    VARCHAR2(10),
    Method_ID    VARCHAR2(10),
    FOREIGN KEY (Service_Type) REFERENCES Service(Service_Type),
    FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID),
    FOREIGN KEY (Method_ID) REFERENCES Payment_Method(Method_ID));

-- VENMO table ---
CREATE TABLE Venmo (
    V_Method_ID VARCHAR2(10) PRIMARY KEY,
    Username    VARCHAR2(50),
    FOREIGN KEY (V_Method_ID) REFERENCES Payment_Method(Method_ID));

-- ZELLE table ---
CREATE TABLE Zelle (
    Z_Method_ID VARCHAR2(10) PRIMARY KEY,
```

```
    Email       VARCHAR2(100),
    Phone       VARCHAR2(20),
    FOREIGN KEY (Z_Method_ID) REFERENCES Payment_Method(Method_ID));

-- ACTIVITY table ---
CREATE TABLE Activity (
    Activity_ID   VARCHAR2(10) PRIMARY KEY,
    Activity_Date DATE,
    Event_Type    VARCHAR2(50));

-- GENERAL_MEETING table ---
CREATE TABLE General_Meeting (
    Activity_ID     VARCHAR2(10),
    Activity_ID_M   VARCHAR2(10),
    Location        VARCHAR2(100),
    Campaign_Phase  VARCHAR2(50),
    PRIMARY KEY (Activity_ID, Activity_ID_M),
    FOREIGN KEY (Activity_ID) REFERENCES Activity(Activity_ID));

-- GUEST_SPEAKER table ---
CREATE TABLE Guest_Speaker (
    Guest_ID    VARCHAR2(10) PRIMARY KEY,
    Guest_Name  VARCHAR2(100),
    Guest_Email VARCHAR2(100),
    Guest_Phone VARCHAR2(20));

-- WORKSHOP table ---
CREATE TABLE Workshop (
    Activity_ID_W VARCHAR2(10),
    Guest_ID      VARCHAR2(10),
    Topic         VARCHAR2(100),
    PRIMARY KEY (Activity_ID_W, Guest_ID),
    FOREIGN KEY (Activity_ID_W) REFERENCES Activity(Activity_ID),
    FOREIGN KEY (Guest_ID) REFERENCES Guest_Speaker(Guest_ID));
```