

# MIST 4610- Group Project 1

Team Name

Group\_21479\_G9

Members

Chase Bell (<https://github.com/cbello1618> )

Megan Hanson (<https://github.com/megannhanson> )

Garret Davis (<https://github.com/gdavis74343>)

Abhi Vadlamudi (<https://github.com/abhi33ravens>)

Problem Description

Our client is seeking the development and implementation of a system in order to organize their members, events, reservations, and various other parts of their Country Club. The country club is centered around its prestigious Golf course, fancy clubhouse, and various other facilities. The club is well known for its high level golf tournaments, dining celebrations, and pool/spa amenities. The club has found themselves having a difficult time managing member data as well as event times and facility upkeep.

Our client has asked us to develop a database that can achieve solutions to the problems outlined above. In order to carry out this task, we have decided to create a comprehensive data model centered around relational databases in SQL. By doing so, our client will be able to have a 360 view of their center of operations, and effectively manage member data, event scheduling, and facility management.

Due to the prestigious level that this Country Club holds, it is crucial that our client has the necessary resources to effectively and efficiently manage its members. This database will also provide our client with a competitive advantage, as they will have the ability to configure and analyze data in ways that most of their competitors cannot.

We have attached screenshots of our conversation with our client (ChatGPT), which provided us with entities and relationships for a golf club with the parameters we provided:



You

create a database for a client managing a country club with entities Members, Guests, Membership tiers, Staff, Events, Facilities, Amenities, and reservations, and then events and facilities will have an associative entity for many to many relationship, members and guests will have an associative entity for many to many relationship, and members and membership tiers will have an associative entity for many to many relationship,



ChatGPT

Sure, here's a conceptual layout for the database schema based on the requirements you provided:

1. **Members**

- member\_id (Primary Key)
- name
- email
- phone
- address

2. **Guests**

- guest\_id (Primary Key)
- name
- email
- phone
- address

3. **MembershipTiers**

- tier\_id (Primary Key)
- tier\_name
- description
- price

4. **Members\_MembershipTiers** (Associative entity for many-to-many relationship between Members and MembershipTiers)

- member\_id (Foreign Key referencing Members)
- tier\_id (Foreign Key referencing MembershipTiers)

5. **Staff**

- staff\_id (Primary Key)
- name
- role
- email



Message ChatGPT...



6. **Events**

- event\_id (Primary Key)
- title
- description
- date
- time
- location

7. **Facilities**

- facility\_id (Primary Key)
- name
- description
- location

8. **Amenities**

- amenity\_id (Primary Key)
- name
- description

9. **Reservations**

- reservation\_id (Primary Key)
- member\_id (Foreign Key referencing Members)
- guest\_id (Foreign Key referencing Guests)
- event\_id (Foreign Key referencing Events)
- facility\_id (Foreign Key referencing Facilities)
- date
- time
- status (e.g., confirmed, canceled)

10. **Events\_Facilities** (Associative entity for many-to-many relationship between Events and Facilities)

- event\_id (Foreign Key referencing Events)
- facility\_id (Foreign Key referencing Facilities)

11. **Members\_Guests** (Associative entity for many-to-many relationship between Members and Guests)

- member\_id (Foreign Key referencing Members)
- guest\_id (Foreign Key referencing Guests)



This database schema should allow you to efficiently manage a country club's members, guests,

Message ChatGPT...



## Our Data Model

Our data model is designed to effectively integrate, store, and analyze data that is necessary to the core operations of our clients Country Club. The model begins with the Members entity, which is the heart of this Country Club and is what provides profit and development for our client.

The **Members' entities** consist of an identification number unique to each member, as well as the members first and last name, their phone number, street address, and method of payment. Each member can bring in many guests. Members can make many reservations for different events and facilities. **The Guests entity** outlines important information for each guest that comes to visit the country club. The entity assigns a unique ID number to each guest, as well as outlines the guest's first and last name, phone number, and address. **Members and Guests form a many to many relationship**, as a member can have many guests, and a guest can be connected to many members. This is portrayed by the **Member\_Has\_Geusts** entity.

Method of payment is crucial in understanding how the specific member is funding their country club experience and is extremely important to the profitability of the country club. A member can have one form of direct payment, however this can be changed if the member desires. This brings us to the next entity, this being **Payment Methods**. A payment method has a unique integer ID associated with it, as well as a name, ranging from debit and credit cards to venmo and other online transactional services. A payment method can be associated with many members.

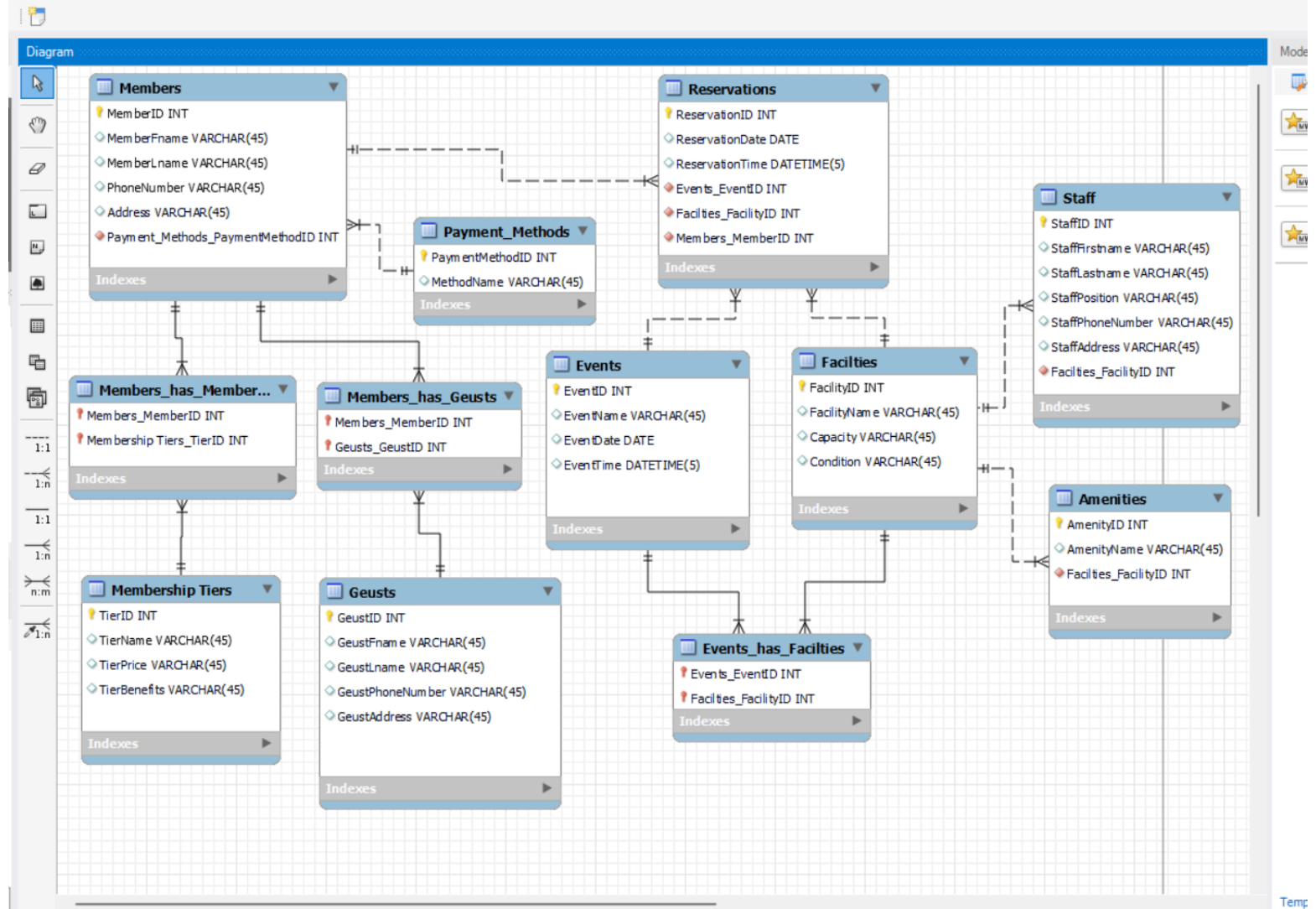
To efficiently and effectively organize their operations, it is important that reservations made by members are kept track of. This is why we created a **reservation entity**. In this entity, reservations are given a unique ID number, as well as a time and date for when the reservation is for. Each reservation occupies a single event and facility, and each reservation entry can only belong to one member.

Next, events are what drive the country club experience for our client, and it is crucial that these events are kept organized. Our **Events entity** gives each event a unique identification number, as well as outlines the event name, event date, and event time. An event can belong to many different reservations made by members. Various events are also part of various different facilities, forming a many to many relationship. This is seen by the **Events\_has\_Facilties** entity. The **Facilities entity** outlines facilities of the country club where various events may be held. This entity consists of a unique facility ID number, facility name, capacity of each facility, and

the condition of the facility. Each facility can be a part of many different events. Furthermore, each facility can hold many reservations. A facility also has many different amenities. **Amenities** are part of facilities, and aid in bringing unique features to the country club, ranging from spas and gyms to swimming pools and tennis courts. The amenity entity supplies each amenity with a unique ID number and the name of the amenity itself.

The staff of the country club is a key part of the operations for our client. **The staff entity** assigns a unique ID number to each staff member, as well as their first name, last name, phone number, address, position, and salary. Each staff member can be a part of one facility, and each facility can have many staff members. This creates a one to many relationship between the two entities.

Finally, Membership tiers are important to assessing what the member has access to at the course and what they are paying to be a part of the country club. **Membership Tiers** entity gives each tier a unique ID number, as well as outlines the tier name and the cost of the tier, which is on a weekly basis of payment. **Members can belong to many memberships, and Membership tiers can belong to many different members, creating a many to many relationship.** This can be seen by the **Members\_has\_MembershipTiers** entity.



## Data Dictionary

**Table 1: Members**

Column Name	Description	Data Type	Size	Format	Key
MemberID	Unique sequential number identifying each member of the golf club	INT			PK
MemberFname	First name of the member	VARCHAR	45		
MemberLname	Last name of the member	VARCHAR	45		
PhoneNumber	Phone number of the member	VARCHAR	45		
Address	Street address of the member	VARCHAR	45		
PaymentMethodID	FK - Payment_Methods; identifies which payment methods are used by the member	INT			FK - Payment_Methods

**Table 2: Reservations**

Column Name	Description	Data Type	Size	Format	Key
ReservationID	Unique sequential number identifying the reservation	INT			PK
ReservationDate	Date of the reservation	DATE		mm-dd-yyyy	
ReservationTime	Time of the reservation on the reservation date	DATETIME	5	hh:mi:ss	
EventID	FK - Events; identifies which event the reservation is associated with	INT			FK - Events
Facility ID	FK - Facilities; identifies which golf club facility the reservation is associated with	INT			FK - Facilities



MemberID	FK - Members; Identifies which member of the golf club the reservation is associated with	INT			FK - Members
----------	--	-----	--	--	--------------

**Table 3: Payment Methods**

Column Name	Description	Data Type	Size	Format	Key
PaymentMethodID	Unique sequential number identifying the payment method	INT			PK
MethodName	Name of the payment method	VARCHAR	45		

**Table 4: Guests**

Column Name	Description	Data Type	Size	Format	Key
GuestID	Unique sequential number identifying the guest	INT			PK
GuestFname	First name of the guest	VARCHAR	45		
GuestLname	Last name of the guest	VARCHAR	45		
GuestPhoneNumber	Phone number of the guest	VARCHAR	45		

GuestAddress	Home address of the guest	VARCHAR	45		
--------------	---------------------------	---------	----	--	--

**Table 5:Events**

Column Name	Description	Data Type	Size	Format	Key
EventID	Unique sequential number identifying the event	INT			<b>PK</b>
EventName	Indicates the name of the event	VARCHAR	45		
EventDate	Indicates the date of the event	DATE		mm-dd-yyyy	
EventTime	Indicates the time of the event on the date of the event	DATETIME		hh:mi:ss	

**Table 6: Facilities**

Column Name	Description	Data Type	Size	Format	Key
FacilityID	Unique sequential number identifying the facility of	INT			PK

	the golf club				
FacilityName	The name of the facility	VARCHAR	45		
Capacity	The maximum number of people (or the capacity) that the facility can hold	VARCHAR	45		
Condition	A brief description of the condition of the facility including recent and planned renovations and issues to be addressed	VARCHAR	45		

**Table 7: Amenities**

Column Name	Description	Data Type	Size	Format	Key
AmenityID	Unique sequential number identifying the amenity item offered by the golf club	INT			PK
AmenityName	Name of the amenity offered by the golf club	VARCHAR	45		
FacilityID	FK - Facilities; indicates				FK - Facilities

	which facility is associated with the amenity				
--	---	--	--	--	--

**Table 8: Membership Tiers**

Column Name	Description	Data Type	Size	Format	Key
TierID	Unique sequential number identifying the membership tier	INT			PK
TierName	Name of the membership tier	VARCHAR	45		
TierPrice	Price required to be paid by the member in order to be in the given tier per year; charged monthly	VARCHAR	45		
TierBenefits	Benefits associated with the tier including access to events, facilities, amenities,	VARCHAR	45		

	and level of priority in making reservations				
--	---	--	--	--	--

**Table 9: Staff**

Column Name	Description	Data Type	Size	Format	Key
StaffID	Unique sequential number identifying the staff member	INT			PK
StaffFirstName	First name of the staff member	VARCHAR	45		
StaffLastName	Last name of the staff member	VARCHAR	45		
StaffPosition	Indicates the name of the position/title the staff member holds	VARCHAR	45		
StaffPhoneNumber	The phone number of the staff member	VARCHAR	45		
StaffAddress	The home address of the staff member	VARCHAR	45		

StaffSalary	The salary for each staff member	INT			
FacilityID	FK - Facilities; indicates the facility in which the staff member works/is responsible for	VARCHAR	45		FK - Facilities

**Table 10: Events\_has\_Facilities**

Column Name	Description	Data Type	Size	Format	Key
EventID	FK - Events; indicates the unique ID number of the event associated with the facility	INT			FK - Events
FacilityID	FK - Facilities; indicates the unique ID number of the facility associated with the event				FK - Facilities

**Table 11: Members\_has\_MembershipTiers**

Column Name	Description	Data Type	Size	Format	Key
MemberID	FK - Members; indicates the unique ID numbers of the members associated with a tier	INT			FK - Members
TierID	FK - Membership Tiers; indicates the unique ID number for the tier associated with a member	INT			FK - Membership Tiers

**Table 12: MemberGeust**

Column Name	Description	Data Type	Size	Format	Key
-------------	-------------	-----------	------	--------	-----

MemberID	FK - Members; indicates the unique ID number of the member associated with each guest	INT			FK - Members
GuestID	FK - Guests; Indicates the unique ID number of the guests associated with each member				FK - Guests

## Queries

	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	Query 7	Query 8	Query 9	Query 10
<b>Multiple Table JOIN</b>		X	X		X		X	X		X
<b>Subquery</b>						X				X
<b>Group BY</b>							X	X	X	X
<b>Group BY with Having</b>								X		X
<b>Multi Condition Where</b>			X	X		X			X	
<b>Built In Functions (count, sum)</b>	X					X	X	X		X
<b>Not Exists/Exists</b>							X			
<b>REGEXP</b>										
<b>Simple</b>	X	X	X	X						

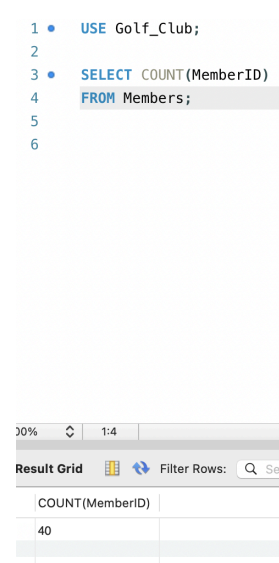


# Queries

## Query 1

This is a simple query that counts the number of members in the golf club. It will allow management to easily see if they have met their goal number of members.

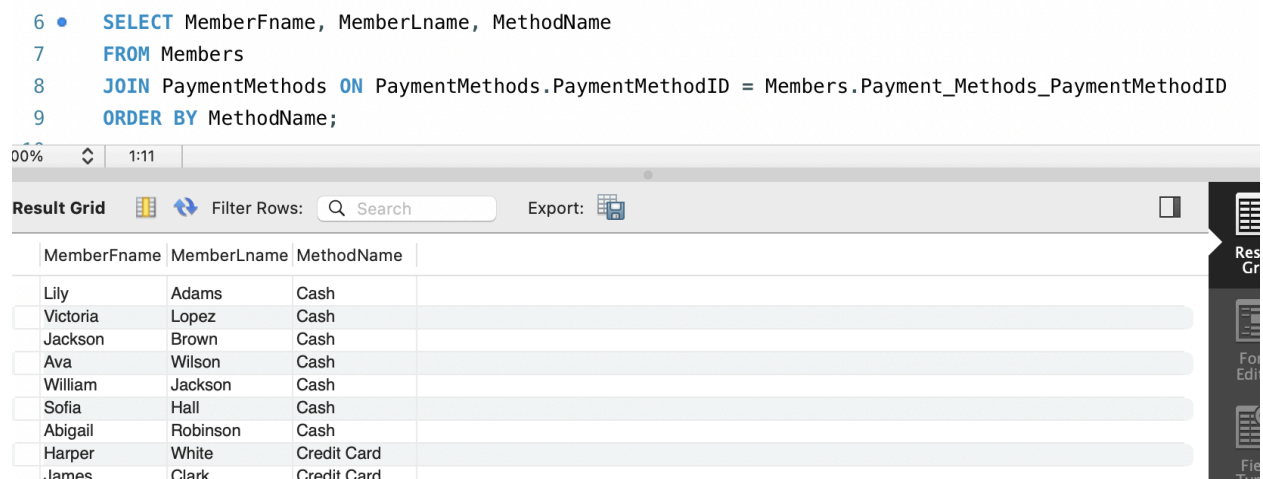
```
SELECT COUNT(MemberID)
FROM Members;
```



## Query 2

This is a simple query including a JOIN and an ORDER BY function. It's used to display the First and Last names of each member along with their associated payment method. It allows management to easily look at the payment method each member uses and groups members together by the method they pay.

```
SELECT MemberFname, MemberLname, MethodName
FROM Members
JOIN PaymentMethods ON PaymentMethods.PaymentMethodID =
Members.Payment_Methods_PaymentMethodID
ORDER BY MethodName;
```



#what is Jaine Kemetz MemberID?

```
SELECT Members.MemberID
FROM Members
JOIN MemberGuest ON MemberGuest.MemberID = Members.MemberID
JOIN Guests ON Guests.GuestID = MemberGuest.GuestID
WHERE GuestFname = "Jaine" AND GuestLname = "Kemetz";
```

	MemberID
▶	6389

```
SELECT Members.MemberID
FROM Members
JOIN MemberGuest ON MemberGuest.MemberID = Members.MemberID
JOIN Guests ON Guests.GuestID = MemberGuest.GuestID
WHERE GuestFname = "Jaine" AND GuestLname = "Kemetz";
```

#### Query 4

#What is the max capacity for the equipmentRoom

```
SELECT Capacity
FROM Facilities
WHERE FacilityName = "EquipmentRoom";
```

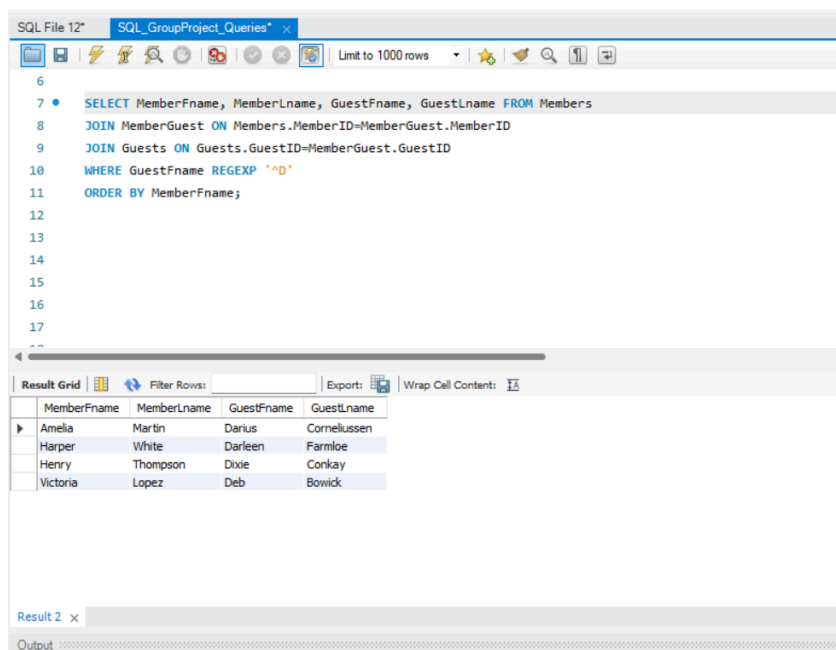
	Capacity
▶	8

```
SELECT Capacity
FROM Facilities
WHERE FacilityName = "EquipmentRoom";
```

### Query 5

This query includes a multiple table JOIN, a WHERE function, and an ORDER BY. It is used to display the names of the guests whose first names start with the letter “D” and the member they are associated with. This will allow management or staff members to easily find the name of a guest and the associated member by inserting the first letter of the guest’s first name.

```
SELECT MemberFname, MemberLname, GuestFname, GuestLname FROM Members
JOIN MemberGuest ON Members.MemberID=MemberGuest.MemberID
JOIN Guests ON Guests.GuestID=MemberGuest.GuestID
WHERE GuestFname REGEXP '^D'
ORDER BY MemberFname;
```



The screenshot shows a SQL query editor window titled "SQL File 12\*" with a tab for "SQL\_GroupProject\_Queries". The query is as follows:

```
6
7 • SELECT MemberFname, MemberLname, GuestFname, GuestLname FROM Members
8 JOIN MemberGuest ON Members.MemberID=MemberGuest.MemberID
9 JOIN Guests ON Guests.GuestID=MemberGuest.GuestID
10 WHERE GuestFname REGEXP '^D'
11 ORDER BY MemberFname;
12
13
14
15
16
17
18
```

Below the query editor, the "Result Grid" is displayed, showing the results of the query. The grid has four columns: MemberFname, MemberLname, GuestFname, and GuestLname. The results are as follows:

MemberFname	MemberLname	GuestFname	GuestLname
Amelia	Martin	Darius	Cornellussen
Harper	White	Darleen	Farmloe
Henry	Thompson	Dixie	Conkay
Victoria	Lopez	Deb	Bowick

At the bottom of the window, there is a "Result 2" tab and an "Output" section.

### Query 6

This query contains a subquery, COUNT clause, and WHERE clause; it is used to display the first and last names and the MemberID of members who have at least one guest associated with them. Members who do not have any guests associated with them are not included in the results.

```
SELECT m.MemberID, m.MemberFname, m.MemberLname,  
(SELECT COUNT(*) FROM MemberGuest mg WHERE mg.MemberID=m.MemberID) AS  
GuestCount  
FROM Members m  
WHERE EXISTS (SELECT 1 FROM MemberGuest mg WHERE  
mg.MemberID=m.MemberID);
```

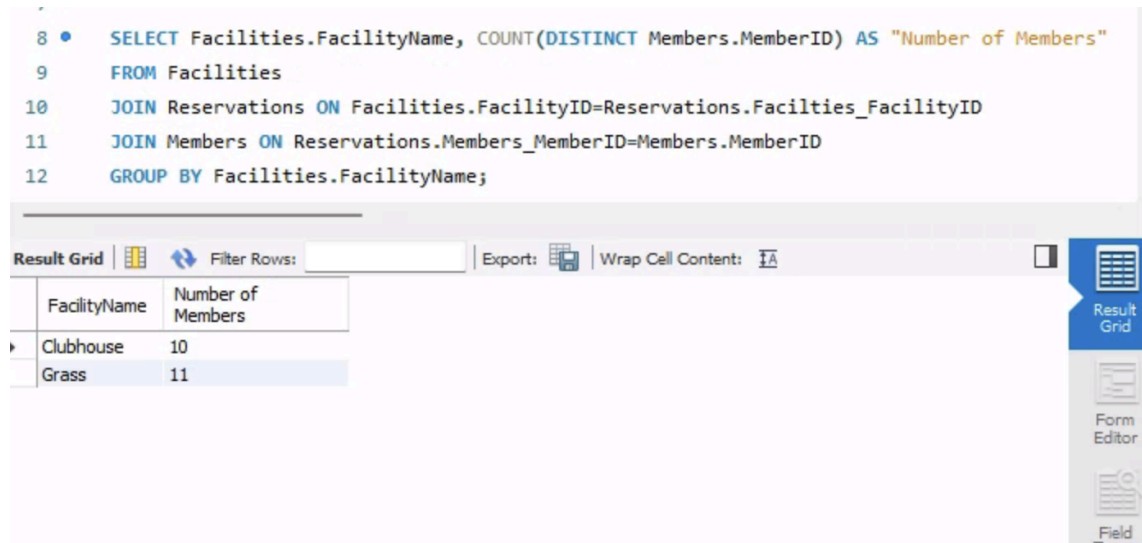
```
1  
2  
3 • SELECT m.MemberID, m.MemberFname, m.MemberLname,  
4 (SELECT COUNT(*) FROM MemberGuest mg WHERE mg.MemberID=m.MemberID) AS GuestCount  
5 FROM Members m  
6 WHERE EXISTS (SELECT 1 FROM MemberGuest mg WHERE mg.MemberID=m.MemberID);
```

MemberID	MemberFname	MemberLname	GuestCount
6389	Madison	Young	1
6804	Emily	Lee	1
6915	Chloe	Gonzalez	1
7163	Emma	Jones	1
7364	Mia	Thomas	1
7640	Samuel	Hill	1
7956	Sofia	Hall	1
8523	Benjamin	Harris	1
8640	Charlotte	Rodriguez	1
8741	Sophia	Johnson	1
8790	Joseph	Green	1
9402	Abigail	Robinson	1
9462	Scarlett	King	1
9675	Elijah	Moore	1

### Query 7

This query contains a multiple table join, a COUNT clause, and a GROUP BY clause. This query is used to display the number of members who have reservations at each facility the golf club has. It will help the golf club staff members to keep track of how many reservations have been made at a facility for a given time frame and help them to plan accordingly.

```
SELECT Facilities.FacilityName, COUNT(DISTINCT Members.MemberID) AS "Number of Members"
FROM Facilities
JOIN Reservations ON Facilities.FacilityID=Reservations.Facilities_FacilityID
JOIN Members ON Reservations.Members_MemberID=Members.MemberID
GROUP BY Facilities.FacilityName;
```



The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

```
8 • SELECT Facilities.FacilityName, COUNT(DISTINCT Members.MemberID) AS "Number of Members"
9 FROM Facilities
10 JOIN Reservations ON Facilities.FacilityID=Reservations.Facilities_FacilityID
11 JOIN Members ON Reservations.Members_MemberID=Members.MemberID
12 GROUP BY Facilities.FacilityName;
```

Below the query, the results are displayed in a table with the following columns: FacilityName and Number of Members. The results are:

FacilityName	Number of Members
Clubhouse	10
Grass	11

The interface also includes a 'Result Grid' button, a 'Form Editor' button, and a 'Field Tunes' button.

### Query 8

This query uses COUNT functions, a multiple table JOIN, and GROUP by in order to display the total amount of revenue generated from the members in each membership tier of the golf club. This is calculated by finding the number of members in each tier, then multiplying the price for the tier by the number of members in the tier. It can be used by the management team to understand the popularity and amount of revenue streams from the different tiers.

```
SELECT MembershipTiers.TierPrice, MembershipTiers.TierName, COUNT(DISTINCT
Members.MemberID) AS "Number of Members",
COUNT(DISTINCT Members.MemberID)*MembershipTiers.TierPrice AS "Total Revenue"
FROM MembershipTiers
JOIN MemberHasTier ON MemberHasTier.TierID = MembershipTiers.TierID
JOIN Members ON MemberHasTier.MemberID = Members.MemberID
GROUP BY MembershipTiers.TierPrice, MembershipTiers.TierName;
```

```
3 • SELECT MembershipTiers.TierPrice, MembershipTiers.TierName, COUNT(DISTINCT Members.MemberID) AS "Number of Members",
4 COUNT(DISTINCT Members.MemberID)*MembershipTiers.TierPrice AS "Total Revenue"
5 FROM MembershipTiers
6 JOIN MemberHasTier ON MemberHasTier.TierID = MembershipTiers.TierID
7 JOIN Members ON MemberHasTier.MemberID = Members.MemberID
8 GROUP BY MembershipTiers.TierPrice, MembershipTiers.TierName;
9
10
```

Result Grid				
Filter Rows:				
Export:				
Wrap Cell Content:				
	TierPrice	TierName	Number of Members	Total Revenue
▶	100	Bronze	8	800
	150	Silver	8	1200
	200	Gold	8	1600
	300	Platinum	8	2400
	400	Diamond	8	3200

### Query 9

This is a complex query using a JOIN, GROUP BY, and a WHERE function. It's used to display the event name, reservation date, reservation time, and reservation ID on the date 6/3/2024. It allows management to easily look at the event they have planned on this specific date 6/3/2024 and make plans accordingly.

```
SELECT EventName, ReservationDate, ReservationTime, ReservationID
FROM Events
JOIN Reservations ON Reservations.Events_EventID = Events.EventID
WHERE ReservationDate = '6/3/2024'
GROUP BY ReservationID;
```

```
11 • SELECT EventName, ReservationDate, ReservationTime, ReservationID
12 FROM Events
13 JOIN Reservations ON Reservations.Events_EventID = Events.EventID
14 WHERE ReservationDate = '6/3/2024';
```

100%

36:14

Result Grid

Filter Rows:

Export:

	EventName	ReservationDate	ReservationTime	ReservationID	
	Dine Fest	6/3/2024	5:30PM	12345	
	Dine Fest	6/3/2024	5:30PM	24680	
	Dine Fest	6/3/2024	5:30PM	54321	
	Dine Fest	6/3/2024	5:30PM	67890	
	Dine Fest	6/3/2024	5:30PM	98765	

### Query 10

Query 10 helps our client further understand the pay structure of their country club. The query analyzes data in order to single out the positions that are currently being paid more than the average staff member, and to evaluate the facilities that they are working at. The query is a complex query using a MULTIPLE TABLE JOIN, A SUBQUERY, A BUILT IN FUNCTION, GROUP BY, AND A HAVING CLAUSE. With this information, our client can further understand if the employees being paid the most are upkeep of their facilities and if their job performance matches their pay rate.

```
SELECT StaffPosition, FacilityName, AVG(StaffSalary) AS 'Average Earnings for Highest Paid Positions'
FROM Staff S
JOIN Facilities F ON F.FacilityID=S.FacilityID
JOIN Events_has_Facilities EF ON EF.Facilities_FacilityID=F.FacilityID
JOIN Events E ON EF.Events_EventID=E.EventID
GROUP BY StaffPosition, FacilityName
HAVING AVG(StaffSalary) > (SELECT AVG(StaffSalary) FROM Staff);
```

The screenshot shows the SQL Server Enterprise Manager interface. The SQL File 12' tab contains the following query:

```
SELECT StaffPosition, FacilityName, AVG(StaffSalary) AS 'Average Earnings for Highest Paid Positions'
FROM Staff S
JOIN Facilities F ON F.FacilityID=S.FacilityID
JOIN Events_has_Facilities EF ON EF.Facilities_FacilityID=F.FacilityID
JOIN Events E ON EF.Events_EventID=E.EventID
GROUP BY StaffPosition, FacilityName
HAVING AVG(StaffSalary) > (SELECT AVG(StaffSalary) FROM Staff);
```

The Result Grid shows the following data:

StaffPosition	FacilityName	Average Earnings for Highest Paid Positions
Host	Clubhouse	69333.3333
Caddy	Grass	86000.0000

The Output pane shows the execution log with the following messages:

- 47 11:36:30 SELECT StaffPosition, FacilityName, EventName, AVG(StaffSalary) AS 'Average Earnings for Highest Paid Positions' 15 row(s) returned
- 48 11:36:41 SELECT StaffPosition, FacilityName, EventName, AVG(StaffSalary) AS 'Average Earnings for Highest Paid Positions' 6 row(s) returned
- 49 11:37:10 SELECT StaffPosition, FacilityName, EventName, AVG(StaffSalary) AS 'Average Earnings for Highest Paid Positions' 6 row(s) returned
- 50 11:37:16 SELECT StaffPosition, FacilityName, AVG(StaffSalary) AS 'Average Earnings for Highest Paid Positions' 2 row(s) returned
- 51 11:38:35 SELECT DISTINCT StaffPosition, FacilityName, EventName, AVG(StaffSalary) AS 'Average Earnings for Highest Paid Positions' 6 row(s) returned
- 52 11:39:33 SELECT StaffPosition, FacilityName, AVG(StaffSalary) AS 'Average Earnings for Highest Paid Positions' 2 row(s) returned