Lab # 04: Database Development using MySQL

OBJECTIVES OF THE LAB

This lab aims at the understanding of:

- *An Example SQL Database Management System i.e. MySQL*
- *Using MySQL Command Line in Windows*
- *MySQL Command Categories*
- *Use of MySQL Commands for a Real-World Problem*

ABOUT MYSQL

With over 10 million installations, MySQL is probably the most popular database management system for web servers. Developed in the mid-1990s, it's now a mature technology that powers many of today's most-visited Internet destinations. It's not only free to use but also extremely powerful and exceptionally fast. MySQL is also highly scalable, which means that it can grow with website.

The SQL in MySQL stands for Structured Query Language. It is the standard relational database language and includes features for defining the structure of the data, modifying data in the database and for specifying security constraints. Apart from MySQL, it is used in Oracle and Microsoft SQL Server.

There are three main ways to interact with MySQL: using a command line, via a web interface such as phpMyAdmin, and through a programming language like PHP. In this lab, we'll cover the command line interaction.

USING MYSQL COMMAND LINE IN WINDOWS

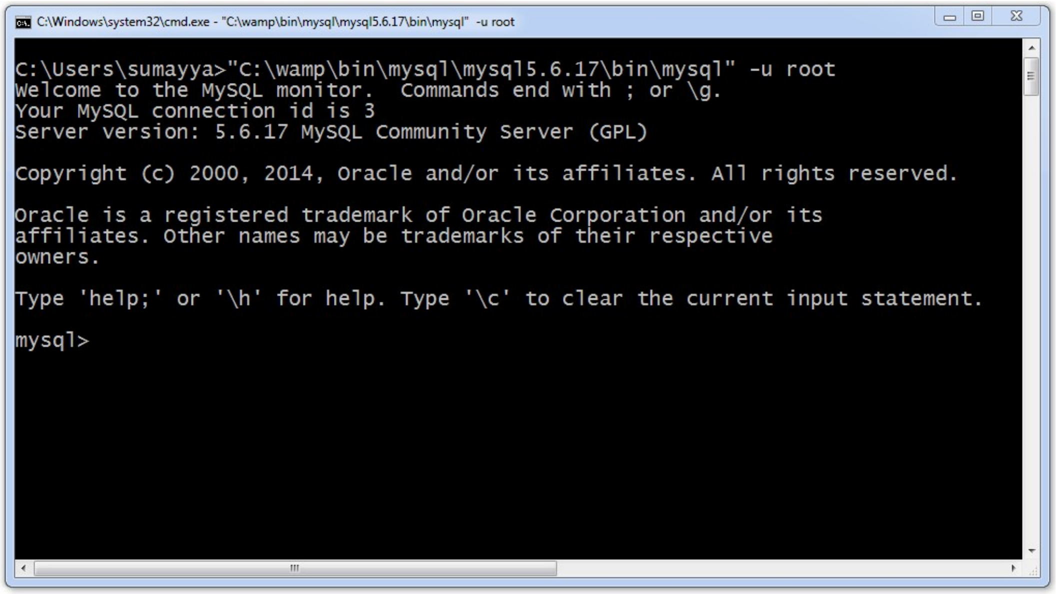
If you installed the WAMP Server, you will be able to access the MySQL executable from the following directory:

- C:\wamp\bin\mysql\mysql5.6.17\bin

By default, the initial MySQL user will be root and will not have had a password set. To enter MySQL's command-line interface, select Start→Run, enter CMD into the Run box, and press Return. This will call up a Windows Command Prompt. From there, enter following:

- "C:\wamp\bin\mysql\mysql5.6.17\bin" –u root

This command tells MySQL to log you in as user root, without a password. You will now be logged into MySQL and can start entering commands as shown in Figure 4.1. Please note that your wamp server must be running, otherwise MySQL command prompt will not show up.


C:\Windows\system32\cmd.exe - "C:\wamp\bin\mysql\mysql5.6.17\bin\mysql" -u root

```
C:\Users\sumayya>"C:\wamp\bin\mysql\mysql5.6.17\bin\mysql" -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

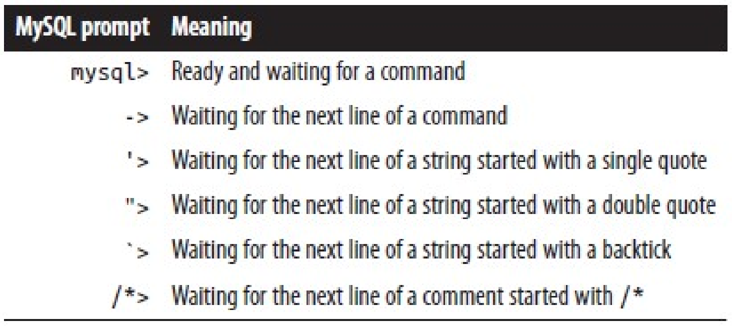
Figure 4.1 – MySQL Command Line Interface

The Semicolon

MySQL uses semicolon to separate or end commands. If you forget to enter it, MySQL will issue a prompt and wait for you to do so. The required semicolon was made part of the syntax to let you enter multiple line commands, which can be convenient because some commands get quite long. It also allows you to issue more than one command at a time by placing a semicolon after each one. The interpreter gets them all in a batch when you press the Enter (or Return) key and executes them in order.

There are six different prompts in MySQL as shown in Table 4.1.

TABLE 4.1 MYSQL COMMAND PROMPTS



MySQL prompt	Meaning
`mysql>`	Ready and waiting for a command
`->`	Waiting for the next line of a command
`'>`	Waiting for the next line of a string started with a single quote
`">`	Waiting for the next line of a string started with a double quote
`` `> ``	Waiting for the next line of a string started with a backtick
`/*>`	Waiting for the next line of a comment started with /*

Note: If you are partway through entering a command and decide you don't wish to execute it after all, whatever you do don't press Control-C! That will close the program. Instead, enter \c and press Return.

MYSQL COMMAND CATEGORIES

SQL statements can be grouped into four general categories: **Data definition language (DDL)**, **Data manipulation language (DML)**, **Transaction control language (TCL)**, and **Data control language (DCL)**. Each one of these is briefly discussed here.

Data Definition Language (DDL)

The overall design of a database is called the database schema. A database schema is specified by a set of definitions that are expressed using a data definition language. It is used for structuring the database. We use the data definition language statements **CREATE**, **ALTER**, **DROP**, **TRUNCATE** to create new objects, alter the structure of existing objects, completely remove objects from system, or delete all rows permanently from the table leaving the structure of the table.

Data Manipulation Language (DML)

These commands are the most frequently used SQL commands. They are used to query and manipulate existing objects like tables. DML commands are: **SELECT**, **UPDATE**, **INSERT**, and **DELETE.**

Transaction Control Language (TCL)

The changes made to the database are known as transaction. Transactions can be made permanent to a database by commit. The various commands in TCL are: **COMMIT**, **SAVEPOINT**, and **ROLLBACK**.

Data Control Language (DCL)

DCL statements are used for securing the database. DCL statements such as **GRANT** and **REVOKE** control access to database and affirm or revoke database transactions. It provides the user with privilege commands. The owner of the database can grant privileges or withdraw (revoke) privilege to other database users so that they can perform the operations accordingly on the tables.

USE OF MYSQL COMMANDS

Consider the department and employee Relational Schema given in Figure 4.2, we will create its MySQL implementation in this section.

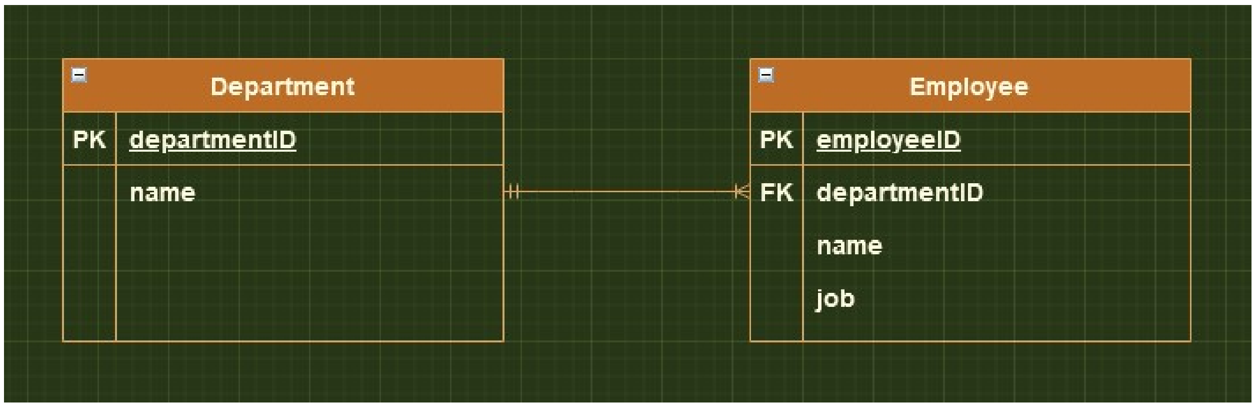


Figure 4.2 – Department-Employee Relational Schema

Command: create database sam_db;
Detail: This command creates a new database sam_db.

Command: show databases;
Detail: This command lists all the existing databases in mysql environment.

Command: use sam_db;
Detail: This command switches over from default mysql database to specified database. In current case, switched database is sam_db.

Command: show tables;
Detail: This command lists all the existing tables in current database i.e. sam_db. Since there are no tables yet in database, so empty set is shown.

Command:
```
create table department(
        departmentID int not null auto_increment,
        name varchar(30),
        primary key(deparmentID));
```

Detail: This command creates a new table department in sam_db database. It consists of two columns departmentID and name. The datatype of each column is specified along with its size. The primary key of table has been set to departmentID attribute. Note that auto_increment option automatically increments the primary key if it's not specified during record entries in table.

Command:

```
create table employee(
        employeeID int not null auto_increment,
        name varchar(80),
        job varchar(30),
        departmentID int default 0,
        primary key(employeeID),
        foreign key(departmentID) references department(departmentID));
```

Detail: This command creates a new table employee in sam_db database. It consists of four columns employeeID, name, job, & departmentID where employeeID has been set as primary key and departmentID as foreign key.

Command:

```
show tables;
```

Detail: This command shows all the existing tables in the sam_db database. Since two tables i.e. department and employee are created in above commands, so these two are listed.

Command:

```
describe department;
```

Detail: This command describes the table structure of department table. Description includes field or attribute names, their respective data types, primary key information, default value of any field and extra options (such as auto_increment).

Command:

```
insert into department values
(10, 'Computer Systems Engineering'),
(15, 'Electrical Engineering'),
(20, 'Chemical Engineering'),
(25, 'Mining Engineering');
```

Detail: This command populates multiple records in department table.

Command:

```
insert into employee values
```

```
(1, 'ABC', 'Lecturer', 10),

(3, 'ACB', 'Lecturer', 10),

(4, 'XYZ', 'Assistant Professor', 10),

(5, 'CAB', 'Lecturer', 15);
```

Detail: This command populates multiple records in employee table.

Command:
```
select * from employee;
```

Detail: This command selects records from employee table where * means to select and show values from all the columns.

Command:
```
select * from employee where job = 'Lecturer';
```

Detail: This command selects all the records from employee table whose job title is 'Lecturer'.

Command:
```
select * from department where departmentID > 15;
```

Detail: This command selects those records from department table whose departmentID is greater than 15.

Command:
```
select * from department where departmentID = 10 or departmentID =20;
```

Detail: This command shows departmental record of those departments whose departmentID is either 10 or 20.

Command:
```
select employee.name, department.name

from employee, department,

where employee.departmentID = department.departmentID;
```

Detail: There are many departments in an organization and a given department contains many employees. We are interested to find employee name and its respective department name, where an employee works. In employee table, department information is shown by departmentID that is a foreign key. Thus, to find accurate department name, only those records must be listed where departmentID in both table matches. This is done by condition specified after where keyword.

Command: select employee.name as empName, department.name as deptName

from employee, department,

where employee.departmentID = department.departmentID;

Detail: This command shows the use of alias feature of MySql. The given command is same as the above one except that column names are specified by alias names i.e. employee name is represented by empName and department name is represented by deptName.

Command: delete from department

where departmentID=25;

Detail: This command deletes a record from department table, whose departmentID is 25.

Command: select * from department;

Detail: This command selects records from <u>department</u> table where * means to select and show values from all the columns. Note that the records shown don't contain department having 25 as it is deleted in above command.

Command: select job from employee;

Detail: This command selects and shows content of 'job' column in employee table.

Command: select distinct job from employee;

Detail: To select different jobs and not the repeated ones, given command is used. Distinct keyword selects only unique jobs present in job column.

Command: select count(job) from employee;

Detail: This command counts the total number of entries in job column of employee table.

Command: select count(distinct job) from employee;

Detail: To count the number of unique jobs only, distinct keyword is used with count function. The result of this command is count of unique jobs in job column.

Command:

```
select count(*) job,

from employee

group by job;
```

Detail: To determine how many employees are hired for a specific job title, following command is used. It finds the count for all the unique jobs.

Command:

```
select count(*) job,

from employee

group by job

having count(*)=1;
```

Detail: This command is the same as above one except the 'having' keyword. Here we are interested to find that *job title*, whose count equals 1 (i.e. find a job for which only one employee is hired?). If such job exists whose employee count is one, then it is returning the count; otherwise an empty set is returned.

Command:

```
select *

from employee

order by job asc;
```

Detail: By default, records are sorted according to primary key in ascending order. But it can be changed. Given command lists all the entries in employee table, such that sorting order of the records is specified by 'job' field.

Command:

```
select *

from employee

limit 3;
```

Detail: To select specific number of records from a given table, limit keyword can be used. The given command selects and displays three records from employee table

Command:

```
update employee

set job='Lab Engineer'

where employeeID=3;
```

Detail: This command updates a record from employee table, sets job to 'Lab Engineer' from 'Lecturer' where employeeID is 3.

So, in this section we have covered the following commands: **create database**, **show databases**, **use**, **create table**, **show tables**, **describe**, **insert**, **update**, **delete**, and **select**.

INTEGRITY CONSTRAINTS USING MYSQL COMMANDS

In this section, we'll apply integrity constraints in created database tables using MySQL commands.

Step 1:

```
alter table employee

drop foreign key employee_ibfk_1;
```

Detail: This command drops the foreign key from the employee table.

Step 2:

```
alter table employee

add foreign key (departmentID) references department(departmentID)

on update cascade;
```

Detail: This command applies the foreign key on the employee table with integrity constraint update cascade. This means that if the foreign key is updated in the parent table i.e. department then it'll be reflected in child table i.e. employee. Similarly, other integrity constraints i.e. on update restrict, on update set null, and on update set default can be applied.

Step 3:

```
update department

set departmentID=12
```

```
where departmentID=10;
```

Detail: This command updates a record from department table, sets departmentID to 12 from 10 where departmentID is 10.

Step 4:

```
select * from department;
```

Detail: This command lists all the entries in the department table. Note that now the departmentID starts from 12 instead of 10 due to previous command.

Step 5:

```
select * from employee;
```

Detail: This command lists all the entries in the employee table. Note that all the employees of Computer Systems Engineering have foreign key value of 12 instead of 10 due to update cascade integrity constraint.

Command:

```
alter table employee

add foreign key (departmentID) references department(departmentID)

on update set null;
```

Detail: This command applies the foreign key on the employee table with integrity constraint update set null. This means that if the foreign key is updated in the parent table i.e. department then it'll be set to null in the child table i.e. employee. Step 1, 3, 4, and 5 are the same as above. Only step has been changed.

---------------------------Task 4.1---------------------------

LAB PERFORMANCE:

1. What is DDL, DML, TCL, and DCL? Explain in your own words. Also, list few commands in each language.
2. **Hands-On:** Do all the commands covered in this lab and show it to the lab instructor.
3. Evaluate yourself and write a note about your understanding of this lab.

---------------------------Task 4.2---------------------------

What is difference between SQL and MySQL? Why is MySQL used? What are its features?

---------------------------Task 4.3---------------------------

What is database engine? What purpose does it serve? How many types of engines are supported by MySQL? Which database engine is most commonly used and why?

---------------------------Task 4.4---------------------------

Specify at least fifteen (15) or more different data types supported by MySQL. Provide the description with at least one example.

---------------------------Task 4.5---------------------------

Consider the Relational Schema given in Figure 4.3 and its tables given in Figure 4.4. Write SQL commands to create all the tables. Take the appropriate attribute type and length from the data provided. (Note: Use the following hierarchy for table creation: 1) Type, Tournament and Team, 2) Member, and 3) Entry).

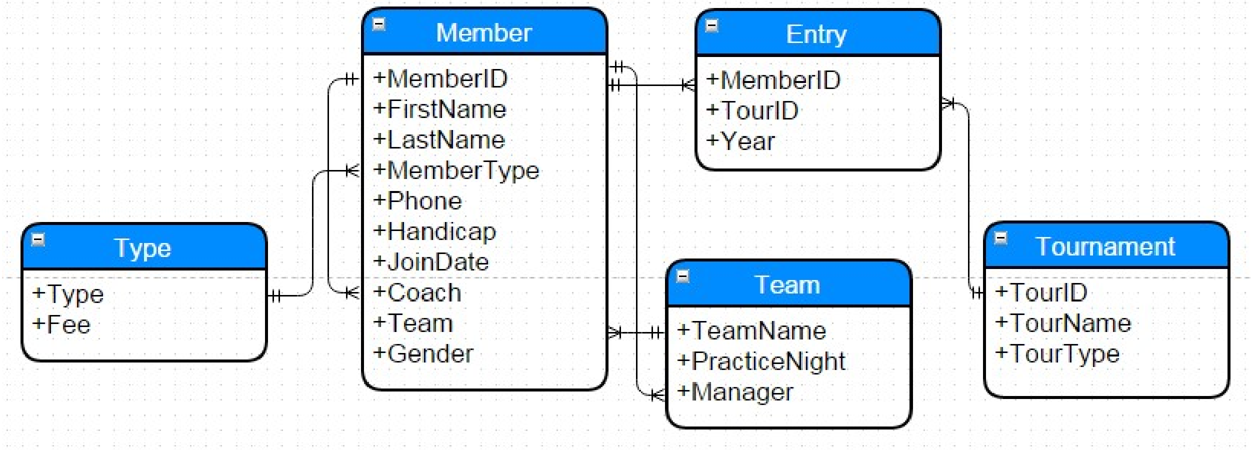


Figure 4.3 – Golf Club Relational Schema

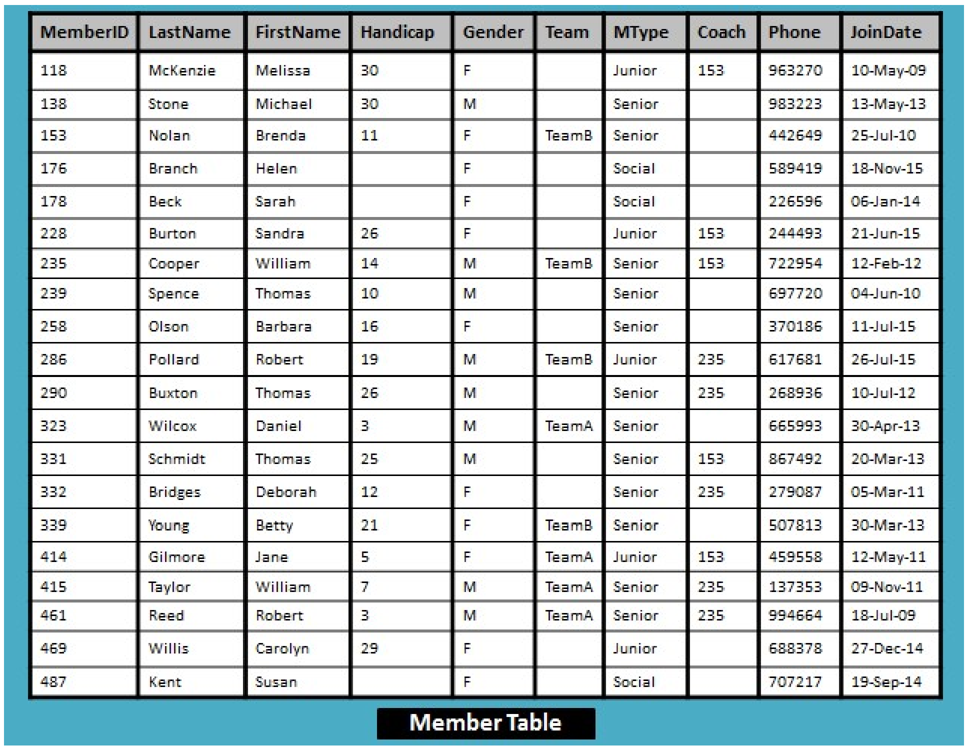
---------------------------Task 4.6---------------------------

Using insert command, populate all the records in member, type, entry, team, and tournament tables according to Figure 4.4a and Figure 4.4b.

---------------------------Task 4.7---------------------------

Write the query for the following:

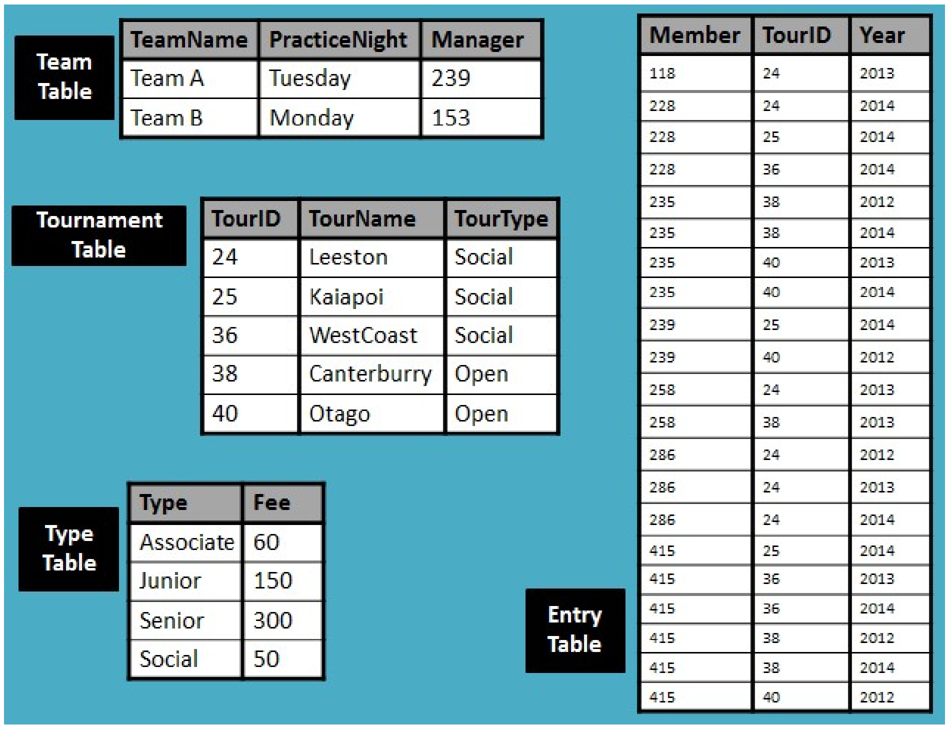
a) List the first name, last name, and phone numbers of all the members.
b) List complete information of all the male members.
c) List complete information of all the members who joined after 01-01-2013.
d) List name of all the members who belonged to Team A.



MemberID	LastName	FirstName	Handicap	Gender	Team	MType	Coach	Phone	JoinDate
118	McKenzie	Melissa	30	F		Junior	153	963270	10-May-09
138	Stone	Michael	30	M		Senior		983223	13-May-13
153	Nolan	Brenda	11	F	TeamB	Senior		442649	25-Jul-10
176	Branch	Helen		F		Social		589419	18-Nov-15
178	Beck	Sarah		F		Social		226596	06-Jan-14
228	Burton	Sandra	26	F		Junior	153	244493	21-Jun-15
235	Cooper	William	14	M	TeamB	Senior	153	722954	12-Feb-12
239	Spence	Thomas	10	M		Senior		697720	04-Jun-10
258	Olson	Barbara	16	F		Senior		370186	11-Jul-15
286	Pollard	Robert	19	M	TeamB	Junior	235	617681	26-Jul-15
290	Buxton	Thomas	26	M		Senior	235	268936	10-Jul-12
323	Wilcox	Daniel	3	M	TeamA	Senior		665993	30-Apr-13
331	Schmidt	Thomas	25	M		Senior	153	867492	20-Mar-13
332	Bridges	Deborah	12	F		Senior	235	279087	05-Mar-11
339	Young	Betty	21	F	TeamB	Senior		507813	30-Mar-13
414	Gilmore	Jane	5	F	TeamA	Junior	153	459558	12-May-11
415	Taylor	William	7	M	TeamA	Senior	235	137353	09-Nov-11
461	Reed	Robert	3	M	TeamA	Senior	235	994664	18-Jul-09
469	Willis	Carolyn	29	F		Junior		688378	27-Dec-14
487	Kent	Susan		F		Social		707217	19-Sep-14

Member Table

Figure 4.4a – Member Table



Team Table

TeamName	PracticeNight	Manager
Team A	Tuesday	239
Team B	Monday	153

Tournament Table

TourID	TourName	TourType
24	Leeston	Social
25	Kaiapoi	Social
36	WestCoast	Social
38	Canterburry	Open
40	Otago	Open

Type Table

Type	Fee
Associate	60
Junior	150
Senior	300
Social	50

Entry Table

Member	TourID	Year
118	24	2013
228	24	2014
228	25	2014
228	36	2014
235	38	2012
235	38	2014
235	40	2013
235	40	2014
239	25	2014
239	40	2012
258	24	2013
258	38	2013
286	24	2012
286	24	2013
286	24	2014
415	25	2014
415	36	2013
415	36	2014
415	38	2012
415	38	2014
415	40	2012

Figure 4.4b – Team, Tournament, Type, and Entry Tables

e) List complete information of all the senior members.
f) List complete information of all the members in order of LastName.

g) Retrieve the number of records in Member table.
h) Provide the first name and last name of the two coaches.
i) Find the amount of fee provided by each member by mentioning member first name, last name, and fee. (Hint: use the member and type tables.)
j) Delete the record from Entry table where Member=415 and TourID=40.
k) Update the Fee of Associate in Type table from 60 to 80.

---------------------------Task 4.8---------------------------

MySQL supports various built-in functions belonging to various categories such as numeric functions, string functions, and date & time functions. Write MySQL commands for following numeric functions: ceiling, cos, degrees, log10, mod, radians, round, sqrt, and truncate. Next write MySQL commands for following string functions: concat, upper, lower, repeat, reverse, regexp, replace, length, ltrim, and rtrim. Finally write MySQL commands for following date & time functions: curdate, week, date_from, quarter, now, sysdate, and date_format.

---------------------------Task 4.9---------------------------

MySQL uses various operators such as Comparison (<, >, <=, >=, ==, and !=), Boolean (AND, OR, and NOT), and Special Operators (Between, Like, IN, Is Null, and Distinct). Give examples of these for Golf database created in this lab.

---------------------------Task 4.10-------------------------

Alter is an important command of MySQL. It is used to alter variety of things associated with a database. It can alter the overall characteristics of database, metadata, view, function, procedure, event, and user. Alter table is used specifically for altering the table metadata. Write MySql statements involving alter table for following:

a) **Add** new **column** **DOB** to store member date of birth. Its type is date and can be null.
b) Now **change** the name of newly added **column** from DOB to **M_DOB** with date as data type and not null.
c) Now **drop** the **M_DOB** **column** from member table.
d) Next **drop** the **primary key** **TourID** from tournament table.
e) Now **add** new **primary key** **TourID** into tournament table.
f) Next **drop** the **foreign key** **Coach** from member table.
g) Now **add** the new **foreign key** **Coach** from member table.

IMPORTANT NOTE ABOUT LAB TASKS

1. Task 4.1 is mandatory to submit during the lab timing as per directions to the lab instructor.
2. It is recommended to complete Task 4.5 and Task 4.6 during the lab. But if unable to complete it then all the remaining tasks must be submitted in the start of week 5 lab. Late submission will not be accepted and awarded zero marks. Showing of output in browser is mandatory.
3. Avoid the use of ChatGPT for answering purpose.