**DATABASE DEVELOPMENT**

**USING MYSQL**

**LAB # 0****4**

**Spring 2025**

**CSE-403L**

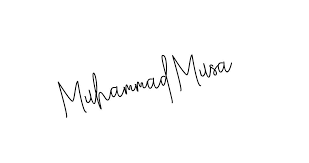
**Database Management System Lab**

Submitted by: **Muhammad Musa**

Registration No.: **22PWCSE2157**

Class Section: **C**

“I affirm that I have completed this work with integrity”



Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Sumayyea Salahuddin**

Monday, March 10, 2025

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Database Management Systems Lab**

# LAB ASSESSMENT RUBRICS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dimension** | **Exemplary** | **Acceptable** | **Developing** | **Unsatisfactory** | **Student**  **Score out of 10**  **Marks** |
| **10** | **8** | **6** | **4** |
| **Overall**  **Impression of**  **Lab Report** | Report is complete, well written, and organized appropriately with additional elements that enhance it. There are no spellings or grammar errors. | Report is complete, briefly written, and organized. There are few spellings and/or grammar errors. | Report is mostly complete, loosely written, and fairly organized. There are spellings and/or grammar errors. | Report is incomplete, sloppy and/or disorganized. There are many spellings and grammar errors that affect clarity. |  |
| **Submission** | Report is submitted on time. | Report is submitted within 24 hours of due date. | Report is submitted within 72 hours of due date. | Report was more than 3 days overdue. |  |
| **Specification** | Programs work and exceed specifications. | Programs work and meet all specifications. | Programs work and meet partial specifications. | Programs work but fail to meet any specifications. |  |
| **Verbal**  **Communication and**  **Understanding** | Answered clearly and accurately with sufficient knowledge. | Answered clearly and accurately with average knowledge. | Answered somewhat clearly and somewhat accurately with limited knowledge. | Answered wrongly and inaccurately with no knowledge. |  |
| **Output Figures/Graphics** | All the output figures and graphics are shown clearly and labeled. | Most of the output figures and graphics are shown clearly and labeled. | Few of the output figures and graphics are shown clearly and labeled. | Output figures and graphics are shown clearly and not labeled. |  |

**Marks**: (\_\_\_\_\_\_\_+\_\_\_\_\_\_\_+\_\_\_\_\_\_\_+\_\_\_\_\_\_\_)/5= \_\_\_\_\_\_\_

**Teacher Remarks and Signature**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Database Development using MySQL**

**Objectives:**

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

**Lab Tasks**:

**Task 4.1**: **LAB PERFORMANCE.**

* **What is DDL, DML, TCL, and DCL? Explain in your own words. Also, list few commands in each language.**

These commands form the basis of SQL operations, enabling the creation, management, manipulation, and security of the data in relational databases.

DDL (Data Definition Language): These commands are used to define or modify the structure of the database itself, such as creating, altering, and dropping tables.

Examples:

*CREATE* to create a new table or database.

*ALTER* to modify an existing database object.

*DROP* to delete an entire table or database.

*DML* (Data Manipulation Language): These commands deal with the manipulation of data present in the database.

Examples:

*SELECT* to retrieve data from the database.

*INSERT* to add new records to a table.

*UPDATE* to modify existing records.

*DELETE* to remove records from a table.

TCL (Transaction Control Language): These are used to manage transactions in a database.

Examples:

*COMMIT* to save the work done.

*ROLLBACK* to undo transactions that are not saved to the database.

*SAVEPOINT* to create points within groups of transactions in which to rollback.

DCL (Data Control Language): These commands are used to control access to data in the database. They are used to grant and revoke rights and permissions.

Examples:

*GRANT* to give users access privileges to the database.

*REVOKE* to remove access privileges.

* **Hands-On: Do all the commands covered in this lab and show it to the lab instructor.**

I have gone throw all the commands and are attached with the lab report in the same zip file

* **Evaluate yourself and write a note about your understanding of this lab.**

Having completed the lab exercises, my understanding of MySQL and its operation in a Windows environment has notably improved. I am now comfortable navigating the MySQL Command Line Interface, which is crucial for directly interacting with databases.

**Task 4.2**: What is difference between SQL and MySQL? Why is MySQL used? What are its features?

SQL is a standard language for accessing and manipulating databases, while MySQL is a database management system that uses SQL to manage the data stored within its databases. MySQL is used because it is open-source, efficient, reliable, and supports large databases. Features of MySQL include its cross-platform support, high performance, strong data protection mechanisms, comprehensive transactional support, and the ability to handle large volumes of data with complex querying capabilities.

**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?

A database engine is the underlying software component that a database management system uses to create, read, update, and delete data from a database. It serves the purpose of handling data storage, manipulation, and retrieval.

MySQL supports several types of engines, including InnoDB, MyISAM, Memory, CSV, and Archive, among others. InnoDB is the most commonly used engine because it supports ACID-compliant transactions, foreign keys for referential integrity, and row-level locking for improved performance with concurrent data access.

**Task 4.4**: Specify at least fifteen (15) or more different data types supported by MySQL. Provide the description with at least one example.

1. **INT**: Stores an integer value. Example: **age INT**.
2. **VARCHAR**: Stores variable-length character strings. Example: **name VARCHAR(255)**.
3. **TEXT**: Stores long text strings. Example: **description TEXT**.
4. **DATE**: Stores date values. Example: **birthdate DATE**.
5. **DATETIME**: Stores date and time values. Example: **created\_at DATETIME**.
6. **TIMESTAMP**: Stores a timestamp, automatically updated to the current date and time when the row is modified. Example: **last\_updated TIMESTAMP**.
7. **FLOAT**: Stores floating-point numbers with an approximate precision. Example: **score FLOAT**.
8. **DOUBLE**: Stores double-precision floating-point numbers with more precision than FLOAT. Example: **balance DOUBLE**.
9. **DECIMAL**: Stores exact precision numeric data. Example: **price DECIMAL(10,2)**.
10. **CHAR**: Stores fixed-length character strings. Example: **gender CHAR(1)**.
11. **BOOLEAN**: Stores TRUE or FALSE values, equivalent to TINYINT(1). Example: **is\_active BOOLEAN**.
12. **ENUM**: Stores one value from a list of predefined values. Example: **status ENUM('new', 'pending', 'complete')**.
13. **SET**: Stores a set of strings that can have zero or more values. Example: **features SET('GPS', 'Bluetooth')**.
14. **BLOB**: Stores binary large objects, such as images or files. Example: **file BLOB**.
15. **JSON**: Stores JSON formatted data. Example: **attributes JSON**.
16. **TINYINT**: Stores a very small integer. Example: **level TINYINT**.
17. **SMALLINT**: Stores a small integer. Example: **steps SMALLINT**.
18. **MEDIUMINT**: Stores a medium-sized integer. Example: **followers\_count MEDIUMINT**.
19. **BIGINT**: Stores a large integer. Example: **population BIGINT**.
20. **BIT**: Stores bit-field values, typically used for storing binary data. Example: **permissions BIT(8)**.

**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).

***SQL Commands:***

-- Table creation for 'Type'

CREATE TABLE Type (

Type VARCHAR(50) PRIMARY KEY,

Fee DECIMAL(10, 2)

);

-- Table creation for 'Tournament'

CREATE TABLE Tournament (

TourID INT PRIMARY KEY,

TourName VARCHAR(100),

TourType VARCHAR(50)

);

-- Table creation for 'Team'

CREATE TABLE Team (

TeamName VARCHAR(100) PRIMARY KEY,

PracticeNight VARCHAR(50),

Manager VARCHAR(100)

);

-- Table creation for 'Member'

CREATE TABLE Member (

MemberID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

MemberType VARCHAR(50),

Phone VARCHAR(15),

Handicap INT,

JoinDate DATE,

Coach VARCHAR(100),

Team VARCHAR(100),

Gender CHAR(1),

FOREIGN KEY (MemberType) REFERENCES Type(Type),

FOREIGN KEY (Team) REFERENCES Team(TeamName)

);

-- Table creation for 'Entry'

CREATE TABLE Entry (

MemberID INT,

TourID INT,

Year YEAR,

PRIMARY KEY (MemberID, TourID),

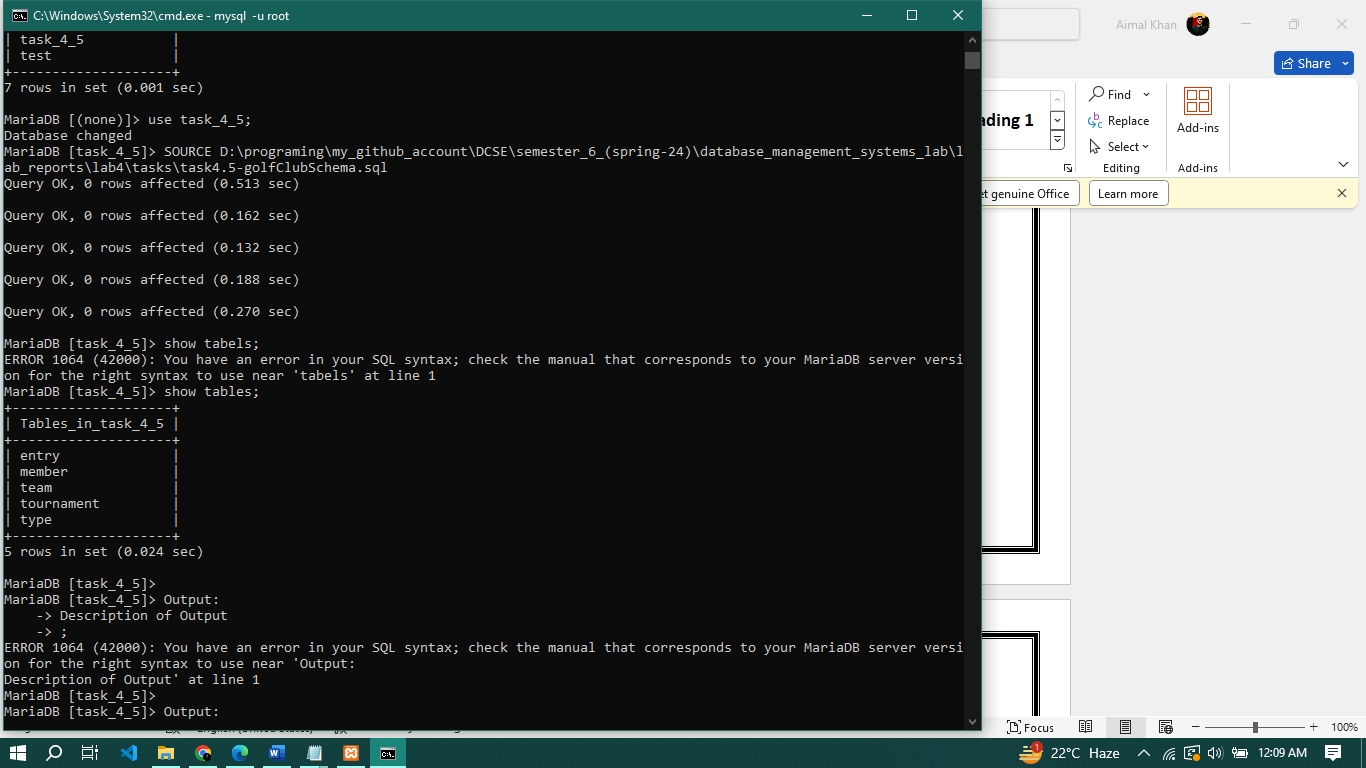
FOREIGN KEY (MemberID) REFERENCES Member(MemberID),

FOREIGN KEY (TourID) REFERENCES Tournament(TourID)

);

***Output:***

After executing the above command, the output is:



**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.

***SQL Command:***

-- Inserting data into the 'Type' table

INSERT INTO Type (Type, Fee) VALUES

('Associate', 60),

('Junior', 150),

('Senior', 300),

('Social', 50);

-- Inserting data into the 'Tournament' table

INSERT INTO Tournament (TourID, TourName, TourType) VALUES

(24, 'Leeston', 'Social'),

(25, 'Kaiapoi', 'Social'),

(36, 'WestCoast', 'Social'),

(38, 'Canterburry', 'Open'),

(40, 'Otago', 'Open');

-- Inserting data into the 'Team' table

INSERT INTO Team (TeamName, PracticeNight, Manager) VALUES

('Team A', 'Tuesday', 239),

('Team B', 'Monday', 153);

-- Inserting data into the 'Member' table

INSERT INTO Member (MemberID, LastName, FirstName, Handicap, Gender, MemberType, Team, Coach, Phone, JoinDate) VALUES

(118, 'McKenzie', 'Melissa', 30, 'F', 'Junior', NULL, 153, 963270, '2009-05-10'),

(138, 'Stone', 'Michael', 30, 'M', 'Senior', 235, NULL, 982732, '2013-05-13'),

(153, 'Nolan', 'Brenda', 11, 'F', 'Senior', 153, NULL, 464249, '2010-07-15'),

(159, 'Branch', 'Helen', 14, 'F', 'Senior', 153, NULL, 482914, '2013-01-15'),

(178, 'Back', 'Sarah', NULL, 'F', 'Social', NULL, NULL, 226559, '2014-06-15'),

(228, 'Button', 'Sandra', 26, 'F', 'Junior', NULL, 153, 244493, '2015-06-21'),

(235, 'Cooper', 'William', 19, 'M', 'Senior', 153, NULL, 722540, '2012-12-01'),

(239, 'Spencer', 'Thomas', 25, 'M', 'Junior', NULL, NULL, 699234, '2011-04-01'),

(258, 'Olson', 'Barbara', 16, 'F', 'Senior', NULL, NULL, 377018, '2014-07-15'),

(286, 'Pollard', 'Robert', 19, 'M', 'Senior', 235, NULL, 616586, '2011-12-15'),

(290, 'Buxton', 'Thomas', 14, 'M', 'Senior', 235, NULL, 268593, '2012-07-15'),

(331, 'Schmidt', 'Deborah', 25, 'F', 'Senior', 153, NULL, 876497, '2013-03-01'),

(339, 'Young', 'Betty', 21, 'F', 'Senior', NULL, NULL, 507813, '2013-03-15'),

(414, 'Gilmore', 'Jane', 5, 'F', 'Social', NULL, NULL, 459558, '2012-01-01'),

(415, 'Taylor', 'William', 7, 'M', 'Senior', 235, NULL, 197536, '2011-09-01'),

(461, 'Reed', 'Robert', 23, 'M', 'Junior', NULL, NULL, 133454, '2014-08-15'),

(469, 'Willis', 'Carolyn', 29, 'F', 'Junior', NULL, NULL, 688378, '2014-12-15'),

(487, 'Kent', 'Susan', 28, 'F', 'Social', NULL, NULL, 707217, '2014-09-15');

-- Inserting data into the 'Entry' table

INSERT INTO Entry (MemberID, TourID, Year) VALUES

(118, 24, 2013),

(228, 24, 2014),

(235, 25, 2014),

(235, 36, 2012),

(235, 38, 2014),

(235, 40, 2012),

(239, 24, 2013),

(239, 25, 2014),

(258, 24, 2013),

(258, 36, 2012),

(286, 24, 2013),

(286, 25, 2012),

(286, 36, 2014),

(286, 38, 2014),

(286, 40, 2012),

(415, 24, 2013),

(415, 25, 2013),

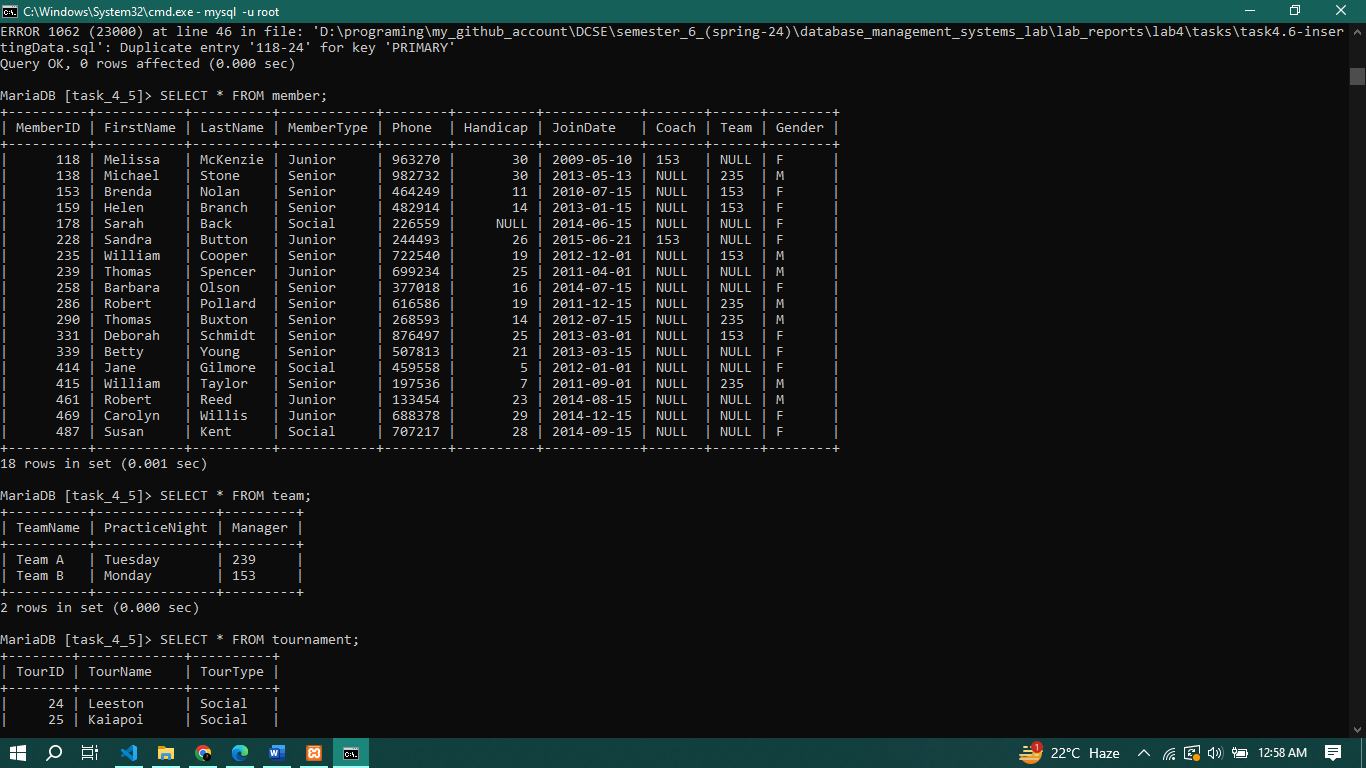
(415, 36, 2014),

(415, 38, 2012),

(415, 40, 2014);

***Output:***

The output after executing the above file as SOURCE path-to/file.sql I got:



**Task 4.7**: Write the query for the following:

* List the first name, last name, and phone numbers of all the members.
* List complete information of all the male members.
* List complete information of all the members who joined after 01-01-2013.
* List name of all the members who belonged to Team A.
* List complete information of all the senior members.
* List complete information of all the members in order of LastName.
* Retrieve the number of records in Member table.
* Provide the first name and last name of the two coaches.
* 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.)
* Delete the record from Entry table where Member=415 and TourID=40.
* Update the Fee of Associate in Type table from 60 to 80.

***SQL Querries:***

-- a) List the first name, last name, and phone numbers of all the members.

SELECT FirstName, LastName, Phone FROM Member;

-- b) List complete information of all the male members.

SELECT \* FROM Member WHERE Gender = 'M';

-- c) List complete information of all the members who joined after 01-01-2013.

SELECT \* FROM Member WHERE JoinDate > '2013-01-01';

-- d) List the name of all the members who belonged to Team A.

SELECT FirstName, LastName FROM Member WHERE Team = 'Team A';

-- e) List complete information of all the senior members.

SELECT \* FROM Member WHERE MemberType = 'Senior';

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

SELECT \* FROM Member ORDER BY LastName;

-- g) Retrieve the number of records in the Member table.

SELECT COUNT(\*) FROM Member;

-- h) Provide the first name and last name of the two coaches. (Assuming coaches' MemberIDs are known, for example, 239 and 153)

SELECT FirstName, LastName FROM Member WHERE MemberID IN (239, 153);

-- i) Find the amount of fee provided by each member by mentioning member first name, last name, and fee. (Assuming the `MemberType` column in `Member` table corresponds to the `Type` in `Type` table)

SELECT M.FirstName, M.LastName, T.Fee

FROM Member M

JOIN Type T ON M.MemberType = T.Type;

-- j) Delete the record from the Entry table where Member=415 and TourID=40.

DELETE FROM Entry WHERE MemberID = 415 AND TourID = 40;

-- k) Update the Fee of Associate in the Type table from 60 to 80.

UPDATE Type SET Fee = 80 WHERE Type = 'Associate';

***Output:***

The output is a bit long. You can run these by saving it in a file having extension ‘.sql’. The run the file as ‘SOURCE path/to/file.sql’ and hit enter you will get all your queries done.

**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.

***Numeric Functions:***

SELECT CEILING(1.7); -- Returns 2

SELECT COS(PI()); -- Returns -1

SELECT DEGREES(PI()); -- Returns 180

SELECT LOG10(100); -- Returns 2

SELECT MOD(11, 4); -- Returns 3

SELECT RADIANS(180); -- Returns PI()

SELECT ROUND(123.4567, 2); -- Returns 123.46

SELECT SQRT(16); -- Returns 4

SELECT TRUNCATE(123.4567, 2); -- Returns 123.45

***String Functions:***

SELECT CONCAT('Hello', ' ', 'World'); -- Returns 'Hello World'

SELECT UPPER('hello'); -- Returns 'HELLO'

SELECT LOWER('HELLO'); -- Returns 'hello'

SELECT REPEAT('MySQL ', 3); -- Returns 'MySQL MySQL MySQL '

SELECT REVERSE('abcd'); -- Returns 'dcba'

SELECT 'abc' REGEXP '^[a-z]$'; -- Returns 1 if 'abc' matches the regular expression

SELECT REPLACE('Hello World', 'World', 'MySQL'); -- Returns 'Hello MySQL'

SELECT LENGTH('Hello World'); -- Returns 11

SELECT LTRIM(' Hello World'); -- Returns 'Hello World' with leading spaces removed

SELECT RTRIM('Hello World '); -- Returns 'Hello World' with trailing spaces removed

***Date and Time Functions:***

SELECT CURDATE(); -- Returns the current date

SELECT WEEK(CURDATE()); -- Returns the week number for the current date

SELECT CURDATE() + INTERVAL 1 DAY; -- Returns the date one day from the current date

SELECT QUARTER(CURDATE()); -- Returns the quarter of the year for the current date

SELECT NOW(); -- Returns the current date and time

SELECT SYSDATE(); -- Returns the current date and time (similar to NOW())

SELECT DATE\_FORMAT(NOW(), '%Y-%m-%d'); -- Returns the current date in YYYY-MM-DD 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.

***Comparison Operators:***

-- Select all members with a handicap less than 20

SELECT \* FROM Member WHERE Handicap < 20;

-- Select all members with a handicap greater than or equal to 30

SELECT \* FROM Member WHERE Handicap >= 30;

-- Select all members who joined before 2014

SELECT \* FROM Member WHERE JoinDate <= '2014-01-01';

-- Select all tournaments where the TourID is not equal to 25

SELECT \* FROM Tournament WHERE TourID != 25;

***Boolean Operators:***

-- Select all male senior members

SELECT \* FROM Member WHERE Gender = 'M' AND MemberType = 'Senior';

-- Select all members who are either in Team A or Team B

SELECT \* FROM Member WHERE Team = 'Team A' OR Team = 'Team B';

-- Select all members who are not juniors

SELECT \* FROM Member WHERE NOT MemberType = 'Junior';

***Special Operators:***

-- Select all members with a handicap between 10 and 20

SELECT \* FROM Member WHERE Handicap BETWEEN 10 AND 20;

-- Select all members whose first name starts with 'Jo'

SELECT \* FROM Member WHERE FirstName LIKE 'Jo%';

-- Select all members who are in either Team A, Team B, or Team C

SELECT \* FROM Member WHERE Team IN ('Team A', 'Team B', 'Team C');

-- Select all members who do not have a coach assigned

SELECT \* FROM Member WHERE Coach IS NULL;

-- Select distinct types of memberships from the Member table

SELECT DISTINCT MemberType FROM Member;

**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:

* Add new column DOB to store member date of birth. Its type is date and can be null.

ALTER TABLE Member ADD DOB DATE NULL;

* Now change the name of newly added column from DOB to M\_DOB with date as data type and not null.

ALTER TABLE Member CHANGE DOB M\_DOB DATE NOT NULL;

* Now drop the M\_DOB column from member table.

ALTER TABLE Member DROP COLUMN M\_DOB;

* Next drop the primary key TourID from tournament table.

ALTER TABLE Tournament DROP PRIMARY KEY;

* Now add new primary key TourID into tournament table.

ALTER TABLE Tournament ADD PRIMARY KEY (TourID);

* Next drop the foreign key Coach from member table.

ALTER TABLE Member DROP FOREIGN KEY Coach;

* Now add the new foreign key Coach from member table.

ALTER TABLE Member ADD CONSTRAINT Coach FOREIGN KEY (Coach) REFERENCES Tournament(TourID);

The End.