**Database management system Lab CSE-403L**

**Semester:6th**



**Lab Report # 6**

**Submitted By:** *Zainab Khalid*

**Registration No:***19PWCSE1743*

**Section: A**

**Submitted to:** *Mam Naina Said*

**Department of Computer Systems Engineering**

**University of Engineering and Technology Peshawar**

**LAB # 06**

**DATABASE DEVELOPMENT USING MYSQL**

**Task 6.1**

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

**Difference Between SQL and MySQL:**

|  |  |
| --- | --- |
| SQL | MySQL |
| SQL is developed by Microsoft Corporation. | MySQL was developed by MySQL AB but is currently acquired and owned by Oracle Corporation. |
| SQL is a query programming language that manages RDBMS. | MySQL is a relational database management system that uses SQL. |
| SQL is primarily used to query and operate database systems. | MySQL allows you to handle, store, modify and delete data and store data in an organized way. |
| SQL does not support any connector. | MySQL comes with an in-built tool known as MySQL Workbench that facilitates creating, designing, and building databases. |
| SQL follows a simple standard format without many or regular updates. | MySQL has numerous variants and gets frequent updates. |
| SQL supports only a single storage engine. | MySQL offers support for multiple storage engines along with plug-in storage, making it more flexible. |
| SQL does not allow other processors or even its own binaries to manipulate data during execution. | MySQL is less secure than SQL, as it allows third-party processors to manipulate data files during execution. |

**Task 6.2**

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?

**Database Engine:**

*A database engine (or storage engine) is the underlying software component that a database management system (DBMS) uses to create, read, update and delete (CRUD) data from a database.*

The Database Engine component of SQL Server is the core service for storing, processing, and securing data. The Database Engine provides controlled access and rapid transaction processing to meet the requirements of the most demanding data consuming applications in your enterprise.

**Types of Storage Engines:**

## There are two types of storage engines in MySQL: transactional and non-transactional. MySQL supported storage engines:

* InnoDB
* MyISAM
* Memory
* CSV
* Merge
* Archive
* Federated
* Blackhole

**Widely Used Storage Engine:**

InnoDB is the most widely used storage engine with transaction support. It is an ACID compliant storage engine. It supports row-level locking, crash recovery and multi-version concurrency control. It is the only engine which provides foreign key referential integrity constraint. Oracle recommends using InnoDB for tables except for specialized use cases.

**Task 6.3:**

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

description with at least one example.

In MySQL, there are various data types that are grouped in numeric (integer, float, boolean, etc.), date and time (DATETIME, DATE, etc.), string (CHAR, VARCHAR, etc.), spatial, and JSON.

The syntax of data types is as follows:

CREATE TABLE table\_name (

column1\_name data type(length),

column2\_name data type(length),

);

**Data Types in MySQL:**

MySQL supports several standard SQL data types. Each column can contain only one data type. In MySQL, data types are grouped in different categories:

1. **Numeric**

* MySQL supports numeric data types such as
* Integers
* Decimals
* Floating-point
* BIT Data Type

1. **Date and time**

For managing date and time information in databases, MySQL date types are used that are divided into:

* DATE
* TIME
* DATETIME
* TIMESTAMP
* YEAR.

1. **String**

The MySQL string data types are divided into:

* TEXT
* BLOB
* CHAR and VARCHAR
* BINARY and VARBINARY
* ENUM
* SET

1. **Spatial**

* GEOMETRY

Stores any type of the geometry value

* POINT

Stores a MySQL single X and Y coordinate value

* LINESTRING

Stores a set of points that form a curve

* POLYGON

Stores a set of points in the multisided geometry

* MULTIPOINT

Stores a set of multiple point values

* MULTILINESTRING

Stores a set of multiple LINESTRING values

* MULTIPOLYGON

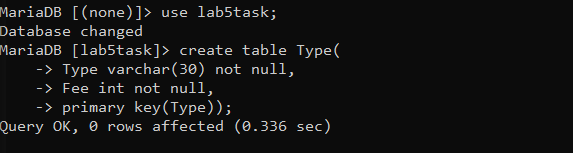
Stores a set of multiple POLYGON values

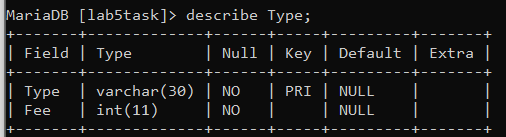
* GEOMETRYCOLLECTION

Stores a set of multiple GEOMETRY values

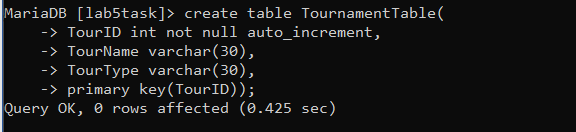
**Task 6.4** :  
Consider the Relational Schema given in Figure 6.2 and its tables given in Figure 6.3. Write SQL commands to create all the tables. Take the appropriate attribute type and length from the data provided. (Note: Use the following hierarchy for table creation: 1) Type, Tournament and Team, 2) Member, and 3) Entry).

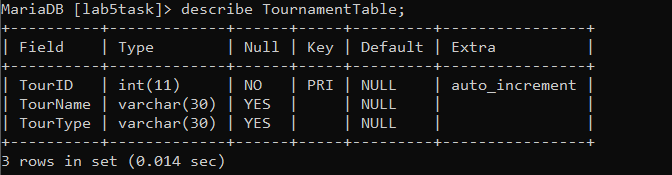
**Type table:**



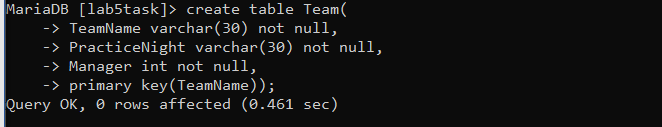


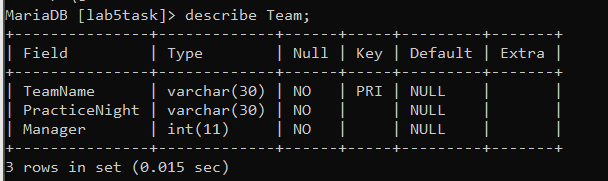
**Tournament table:**



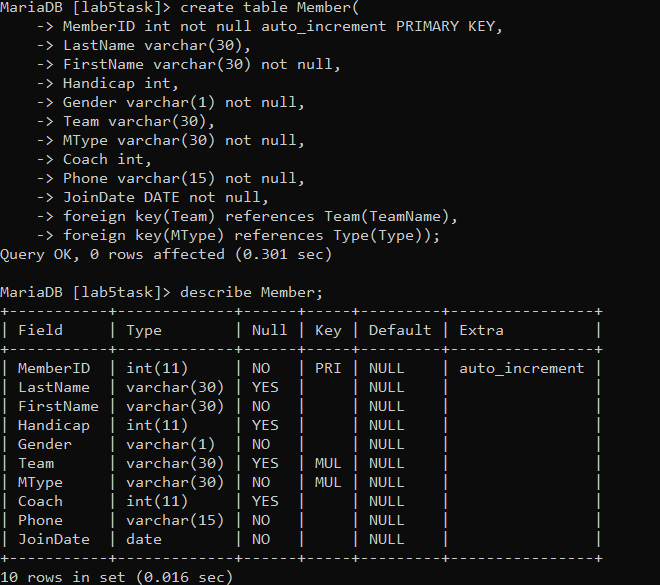


**Team Table**

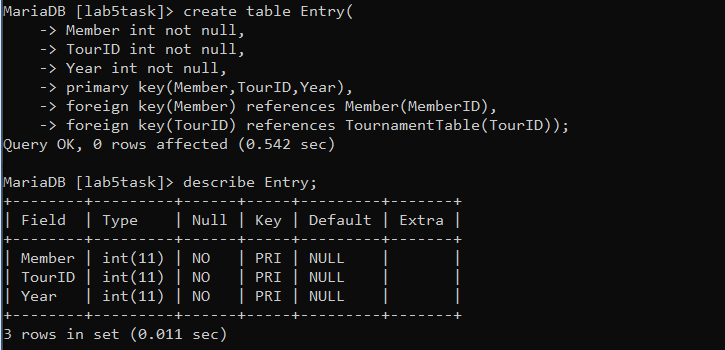




**Member Table:**

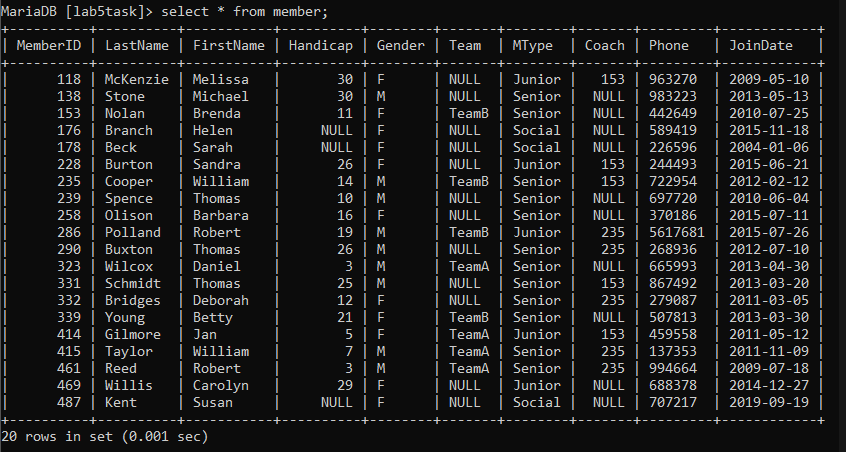


**Entry Table:**

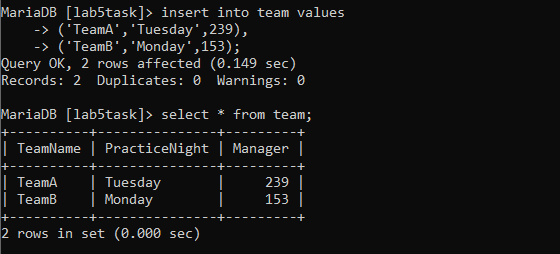


**Task 6.5**Using insert command, populate all the records in member, type, entry, team, and tournament tables according to Figure 6.3a and Figure 6.3b.

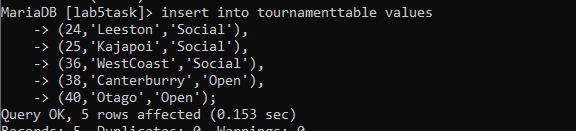
**Member table**

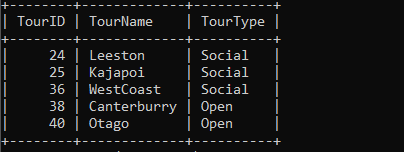


**Team table**

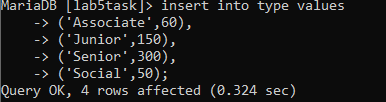


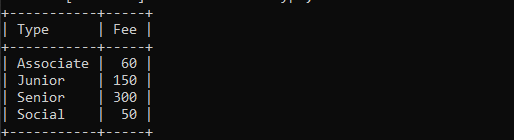
**Tournament table**



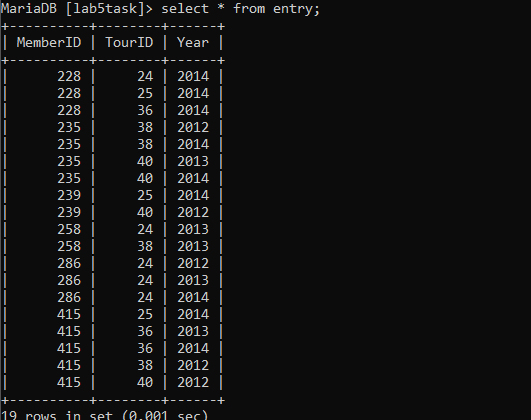


**Type table**





**Entry table**



**Task 6.6:**

Write the query for the following:

1. List the first name, last name, and phone numbers of all the members.

select FirstName, LastName, Phone from members;

1. List complete information of all the male members.

select \* from members where Gender = 'M';

1. List complete information of all the members who joined after 01‐01‐2013.

select \* from members where JoinDate= '01-Jun-13';

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

select \* from members where Team = 'TeamA';

e) List complete information of all the senior members.

select \* from members where MType= 'Senior';

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

select \* from members order by LastName asc;

g) Retrieve the number of records in Member table.

select count(\*) from members

h) Provide the first name and last name of the two coaches.

select FirstName, LastName from members limit 2

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

DELETE FROM member WHERE Member=415 and TourID=40;

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

UPDATE Type

SET Fee= 80;

**Task 6.7**

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:**

**CEILING:**

The CEILING() function returns the smallest integer value that is bigger than or equal to a number.

**Syntax:**

CEILING(*number*)

**Example:**

Return the smallest integer value that is greater than or equal to 25:

SELECT CEILING(25);

**COS:**

The COS() function returns the cosine of a number.

**Syntax**

COS(*number*)

**Example**

Return the cosine of a number:

SELECT COS(2);

**DEGREES:**

The DEGREES() function converts a value in radians to degrees.

**Syntax**

DEGREES(*number*)

**Example**

Convert the radian value into degrees:

SELECT DEGREES(1.5);

**LOG10:**

The LOG10() function returns the natural logarithm of a number to base-10.

**Syntax**

LOG10(*number*)

### **Example**

Return the base-10 logarithm of 4.5:

SELECT LOG10(4.5);

**MOD:**

The MOD() function returns the remainder of a number divided by another number.

**Syntax**

* MOD(*x*, *y*)
* *x* MOD *y*
* *x*% *y*

**Example**

Return the remainder of 18/4:

SELECT 18 MOD 4;

**RADIANS:**

The RADIANS() function converts a degree value into radians.

**Syntax**

RADIANS(*number*)

### **Example**

Convert a degree value into radians:

SELECT RADIANS(-45);

**ROUND:**

The ROUND() function rounds a number to a specified number of decimal places.

**Syntax**

ROUND(*number*, *decimals*)

### **Example**

Round the number to 2 decimal places:

SELECT ROUND(135.375, 2);

**SQRT:**

The SQRT() function returns the square root of a number.

**Syntax**

SQRT(*number*)

### **Example**

Return the square root of a number:

SELECT SQRT(64);

**TRUNCATE:**

The TRUNCATE() function truncates a number to the specified number of decimal places.

**Syntax**

TRUNCATE(*number*, *decimals*)

### **Example**

Return a number truncated to 2 decimal places:

SELECT TRUNCATE(135.375, 2);

**Date & Time Functions:**

**CURDATE:**

The CURDATE () function returns the current date.The date is returned as "YYYY-MM-DD" (string) or as YYYYMMDD (numeric).This function equals the [CURRENT\_DATE()](https://www.w3schools.com/sql/func_mysql_current_date.asp) function.

**Syntax**

CURDATE()

### **Example**

Return the current date:

SELECT CURDATE();

**WEEK:**

The WEEK() function returns the week number for a given date (a number from 0 to 53).

**Syntax**

WEEK(*date*, *firstdayofweek*)

### **Example**

Return the week number for a date:

SELECT WEEK("2017-06-15");

**QUARTER:**

The QUARTER() function returns the quarter of the year for a given date value (a number from 1 to 4).

* January-March returns 1
* April-June returns 2
* July-Sep returns 3
* Oct-Dec returns 4

## Syntax

QUARTER(date)

### **Example**

Return the quarter of the year for the date:

SELECT QUARTER("2017-06-15");

**NOW:**

The NOW() function returns the current date and time.

**Note:** The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric).

**Syntax**

NOW()

### **Example**

Return current date and time:

SELECT NOW();

**SYSDATE:**

The SYSDATE() function returns the current date and time.

**Note:** The date and time is returned as "YYYY-MM-DD HH:MM:SS" (string) or as YYYYMMDDHHMMSS (numeric).

**Syntax**

SYSDATE()

### **Example**

Return the current date and time:

SELECT SYSDATE();

**DATE\_FORMAT:**

The DATE\_FORMAT() function formats a date as specified.

**Syntax**

DATE\_FORMAT(*date*, *format*)

### **Example**

Format a date:

SELECT DATE\_FORMAT("2017-06-15", "%Y");

**TASK 6.8**

MySQL uses various operators such as Comparison (<, >, <=, >=, ==, and !=), Boolean (AND, OR, and NOT), and Special Operators (Between, Like, IN, Is Null, and Distinct). Explore these.

**Comparison Operators:**

**= = (Equal to):**

SELECT \* FROM Products

WHERE Price = 18;

**> (Greater than):**

SELECT \* FROM Products

WHERE Price > 30;

**< (Less than):**

SELECT \* FROM Products

WHERE Price < 30;

**>= (Greater than or equal to):**

SELECT \* FROM Products

WHERE Price >= 30;

**<= (Less than or equal to):**

SELECT \* FROM Products

WHERE Price <= 30;

**<> (Not equal to):**

SELECT \* FROM Products

WHERE Price <> 18;

## The SQL AND, OR and NOT Operators

The AND and OR operators are used to filter records based on more than one condition:

**NOT:**

Displays a record if the condition(s) is NOT TRUE

SELECT \* FROM Customers

WHERE City NOT LIKE 's%';

**OR:**

TRUE if any of the conditions separated by OR is TRUE

SELECT \* FROM Customers

WHERE City = "London" OR Country = "UK";

**AND:**

TRUE if all the conditions separated by AND is TRUE

SELECT \* FROM Customers

WHERE City = "London" AND Country = "UK";

**Special Operators:**

**BETWEEN:**

TRUE if the operand is within the range of comparisons.

SELECT \* FROM Products

WHERE Price BETWEEN 50 AND 60;

**IN:**

TRUE if the operand is equal to one of a list of expressions

SELECT \* FROM Customers

WHERE City IN ('Paris','London');

**LIKE:**

TRUE if the operand matches a pattern.

SELECT \* FROM Customers

WHERE City LIKE 's%';

**ISNULL:**

ISNULL() function lets you return an alternative value when an expression is NULL:

SELECT ProductName, UnitPrice \* (UnitsInStock + ISNULL(UnitsOnOrder, 0))

FROM Products;

**DISTINCT:**

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT column1, column2, ...

FROM table\_name;