CIS108-6 Data Modelling, Management and Governance

**Assessment:** Report (Design) and Implementation

**Student Name**: Prakash

**Student Number**:

Table of Contents

[Database Solution 3](#_Toc122467701)

[ER DIAGRAM 5](#_Toc122467702)

[USE CASE Diagram 6](#_Toc122467703)

[SQL Queries 7](#_Toc122467704)

# Database Solution

Database solution is a system that allows organizations to manage, store, and retrieve data efficiently. It includes both the database management system software that manages the data as well as the physical database, where all the data is stored.

A database solution typically provides feature such as data indexing, modelling. Query optimization, data security, and backup and recovery to help organizations manage and use their data effectively. It can be used for a wide range of applications, including customer relationship management, financial reporting, inventory management and more.

There are many different types of databases. Solutions available including relational databases, no SQL databases. And in memory databases. Each type has its own strengths and its suitable for different types of data and use cases.

In this project this have been used query to face data and to get data e.g., staff data, member data at etc.

Author also used grant query for permission according to their requirement. This has to make use anyone of the database solution like my SQL and NoSQL.

**Sports Club Record System: -**

Sports Club record system is a database or system that is used to track and manage various types of information related to a Sports Club. This may include information on players, teams, schedules, results, finances and more.

Sports Club record systems may include features to manage finances such as tracking membership fees, merchandise sales and other sources of income. They may also include tools for communication, such as email or messaging capabilities for coaches and players to stay in touch.

Overall, Sports Club record system is a valuable tool for sports clubs to menage and track important information related to their operations. It helps clubs stay organized, keep track of, keep performance matrix and make informed decisions to improve their operations and support the success of their players.

**Requirement: -**

In this project author Basic requirement is structured query language.

We may use whole process time primary key or may use at the end with the alter query to give unique ID for every member of the Sports Club.

Another stuff is that we have to be careful for parsing the data and retrieve data when needed.

Author men also used as data manipulation language, which means we can use it for maintaining an already existing database.

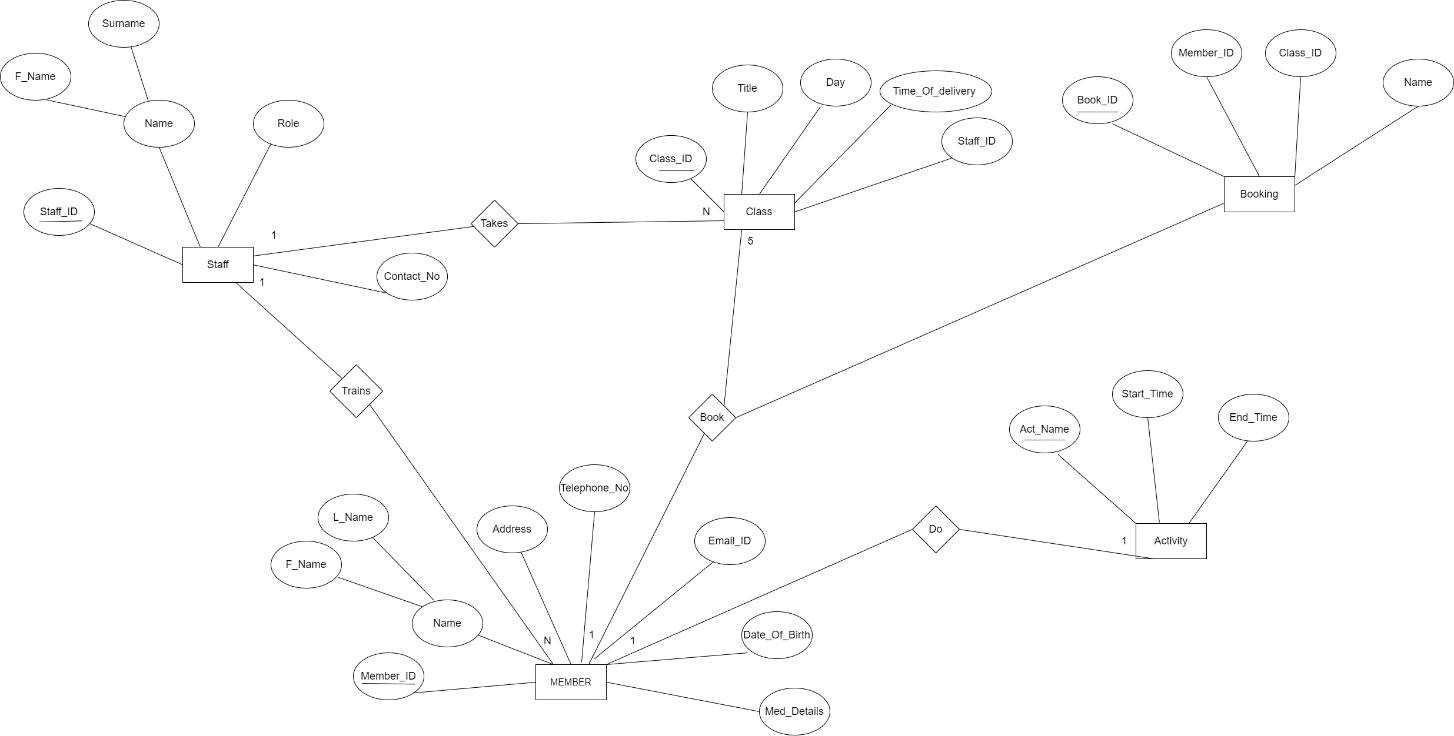
**Advantages of Sports Club Record System: -**

There are several advantages to using a Sports Club record system:

**Improved organization**: A sports club record system helps sports clubs stay organized and keep track of important information related to their operations. This includes player profiles, results, schedules, and finances. **Enhanced performance tracking**: Sports Club record system allows clubs to track and analyses key performance matrix such as scores, statistics and team performance. This helps clubs identify areas for improvement and make informed decisions to support player development. **Increased efficiency**: By automating many of tasks involved in managing a Sports Club, a Sports Club record system can help clubs operate more efficiently disinclined scheduling games and practices, managing finances, and communicating with players and coaches. **Better communication**: A Sports Club record system can provide tools for coaches and players to stand touch, such as email or messaging capabilities. This helps ensure that everyone is on the same page. Can easily share important information. **Cost savings**: By streamlining operation and automating many tasks. Hey, Sports Club. Record system can help sport clubs save time and money. This can allow clubs to allocate their resources more effectively and focus on their area of their operation.

# ER DIAGRAM

ER diagram means Entity Relation diagram. ER diagram is widely used to describe the conceptual design of database. It helps to the client and developer to preview the structure of database before implementing the database. Client easily visualized the data that describes how the data related to each other.



**Challenges I have faced while making ER Diagram:**

To decide weather there is a need to use attributes of one entity as key of another when it is not necessarily mentioned in the statement. Weather to add extra entities which are not there in the problem statement but required for completing the ER. The main challenges faces is incomplete as any other data model. It is important to understand the reason for incompleteness because only then we develop approaches to overcome the model’s limitations.

**Benefits of ERM and how it is beneficial for designing this system:**

ER diagram is the base of any Database. It is very difficult to design a database without ER diagram and contains high risk of issues with design constraints. For the given system it helped both user and developers to preview the structure of the database without actually implement it. It will help remove all the flaws with the database design constraints. This ER diagram can be easily converted into the relational database model.

# USE CASE Diagram:

Use Case diagram also used as graphical depiction of a client, possible interactions with a system.

A Use Case diagram shows various use cases and different types of users the system has and will often be accompanied by others types of diagrams as well.

The Use Case diagram represented by either in the form of circles or ellipses.

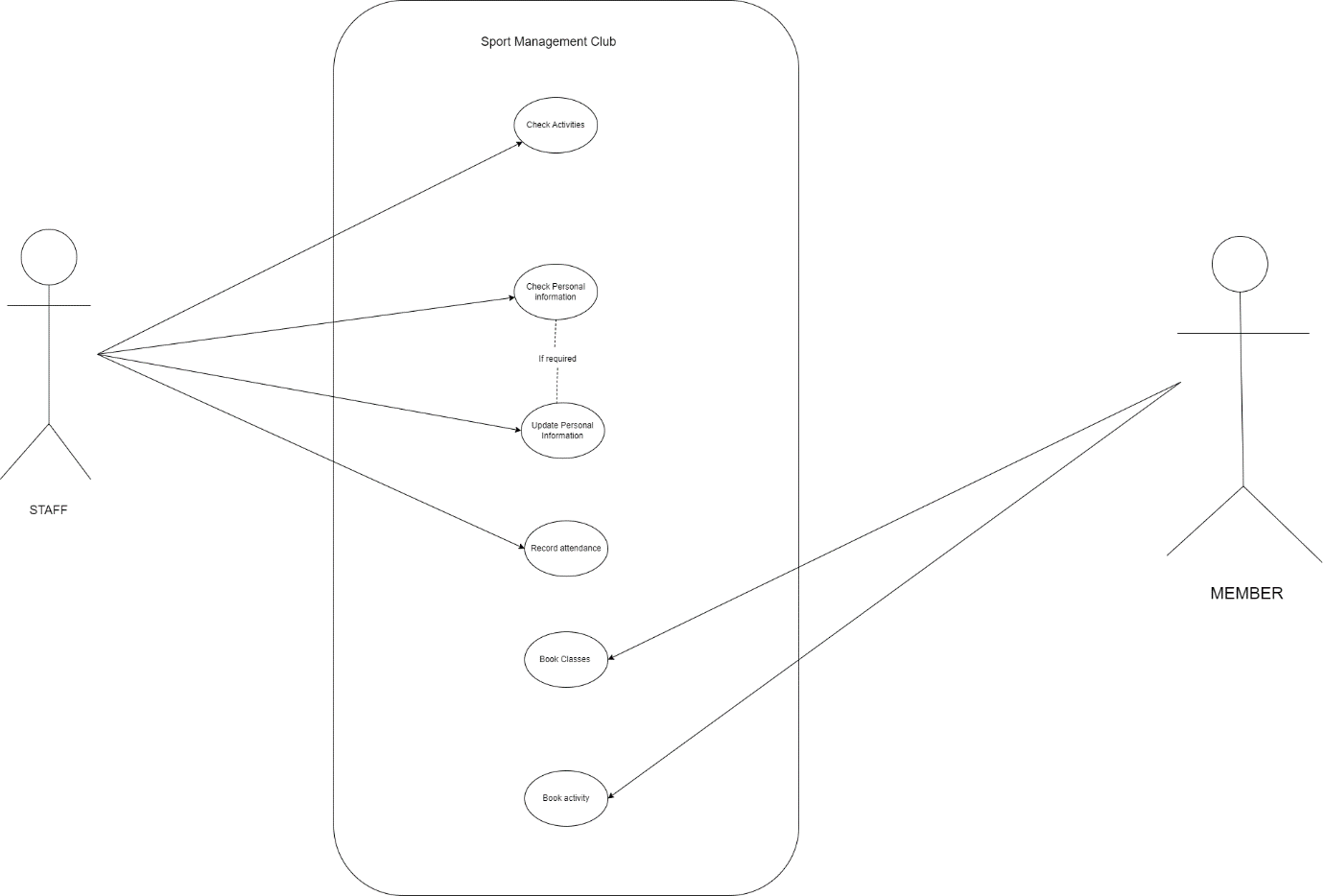
**Benefits of Use Case Diagram:**

Specify system requirements and also represent the goal for system and client.

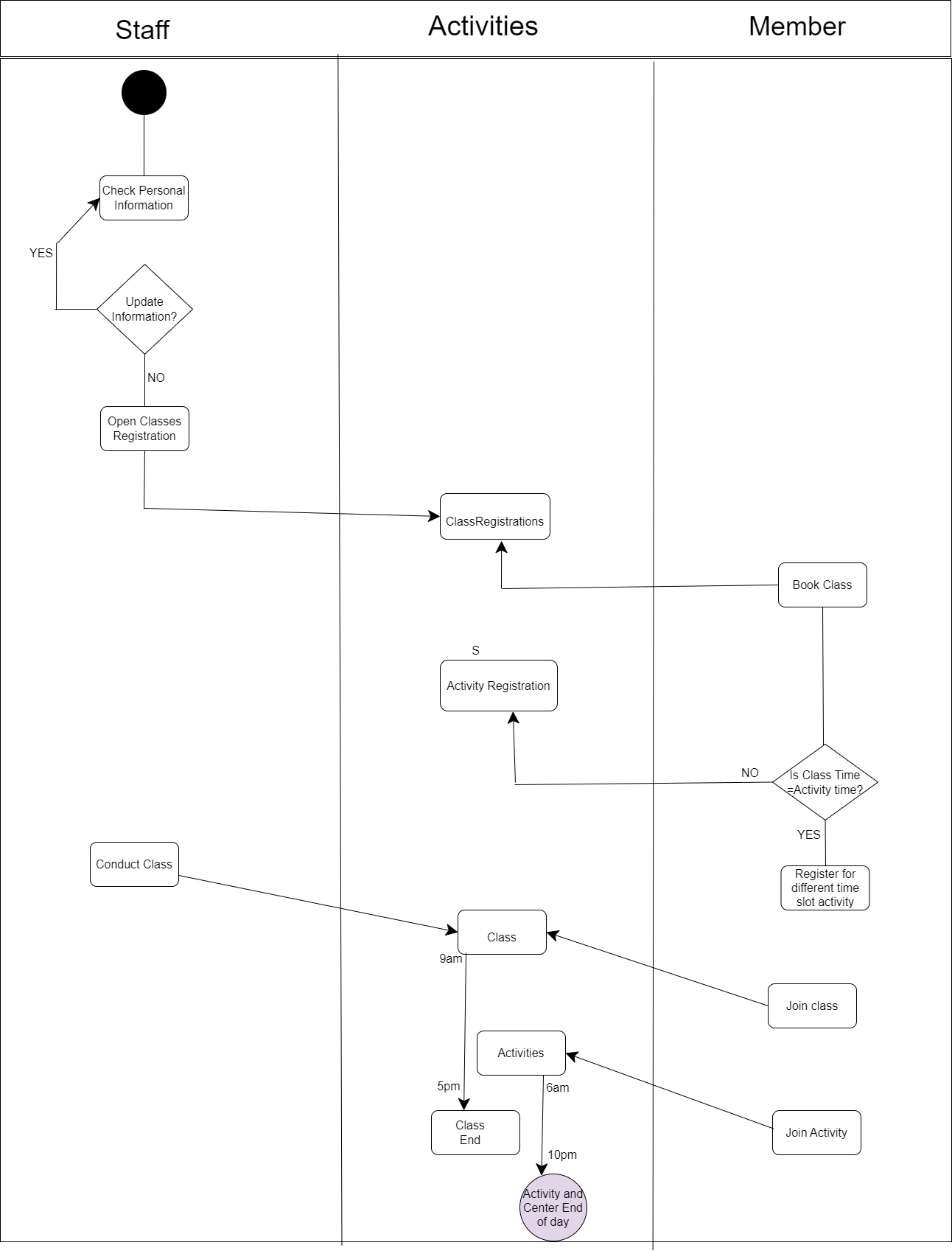
Provide model for the flow of events when it comes to use interactions.

Provides an outside view of a system.

**Use Case Diagram for Sport club record system below**



**Activity Diagram:**



# SQL Queries

SQL-> Structured Query Language -> Used to perform operation on the data and give out the appropriate answer (data)

It helps Developer to access data in the RDBMS (Relational Database Management System).

SQL is used to communicate with the database and it’s a standard language for RDBMS.

SQL statement are used to perform tasks such as update data on a database or retrieve data from a database.

Fives types of Queries author basically used in daily life.

Data Definition Language

Data Manipulation Language

Data Control Language

Transaction Control Language

Data Query Language

**Creating Table for Member’s data: -**

Firstly, create member id which is mandatory i.e. primary key constraint used.

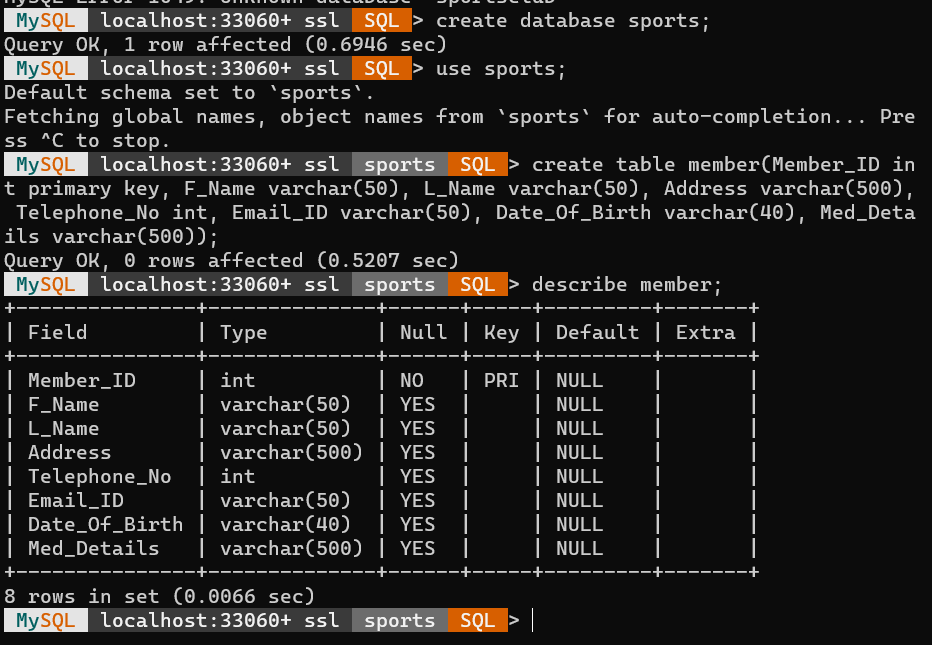
Creating first name and last name using varchar datatype i.e. variable length

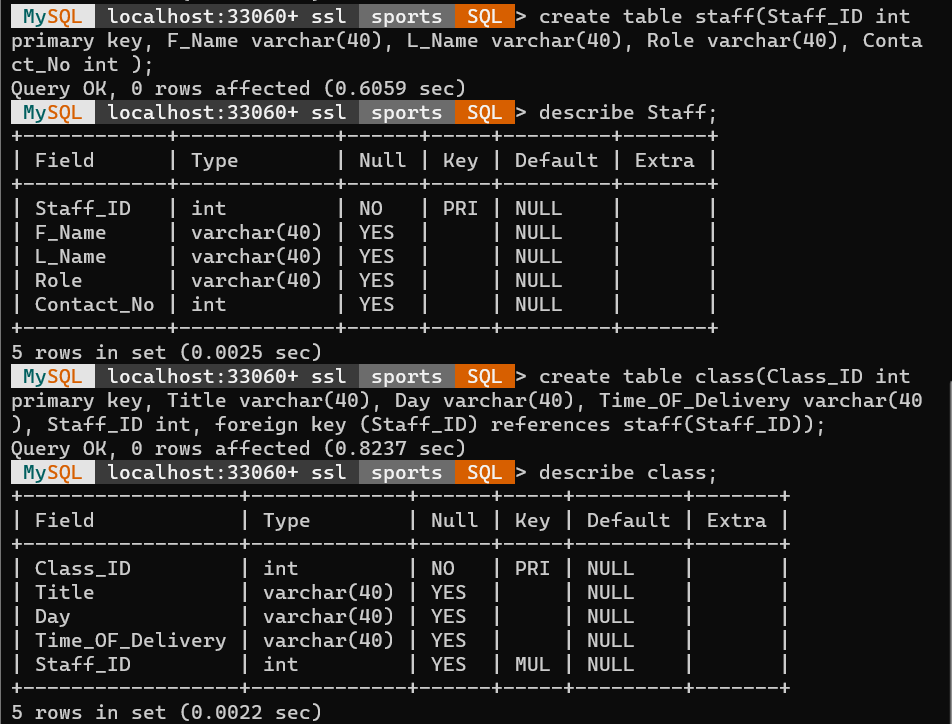
Creating other stuff of the member data e.g.: - address, email id etc.

Member id should be unique so author used primary key to identify each record

Even if you use lower\_case\_table\_names=2 (which enables MySQL to remember the case used for databases and table names), MySQL does not remember the case used for database names for the function [DATABASE()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_database) or within the various logs (on case-insensitive systems).

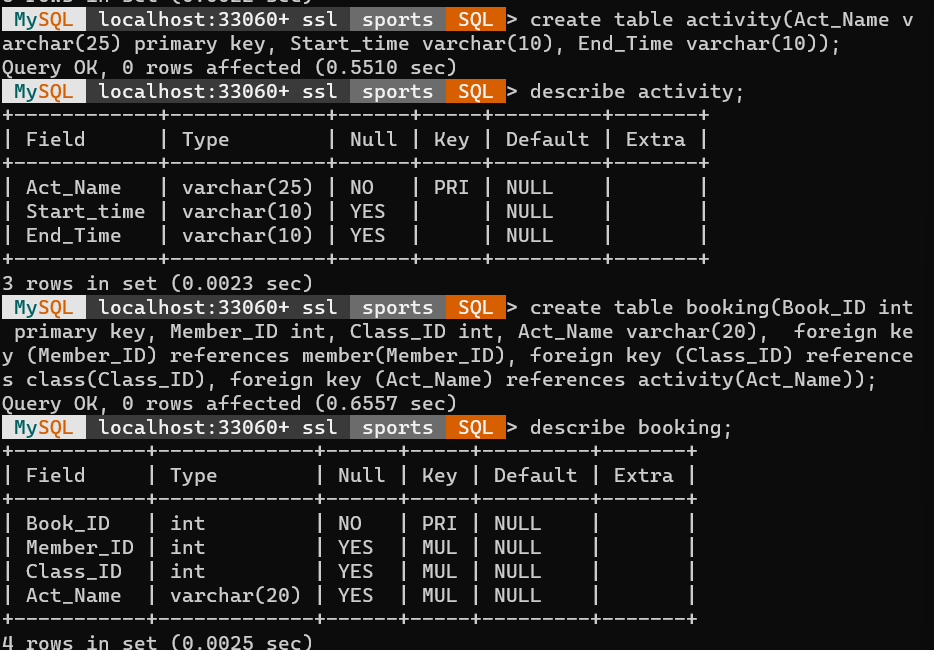
With statement-based binary logging, the source server writes the executed queries to the binary log. This is a very fast, compact, and efficient logging method that works perfectly in most cases. However, it is possible for the data on the source and replica to become different if a query is designed in such a way that the data modification is nondeterministic (generally not a recommended practice, even outside of replication).





**Creating Table for Activity data**

Creating data member for activity table. Number one is activity name that should be unique. After that define other stuff data member with appropriate data type



Challenges Faced

**Creating Table for Member’s data: -**

Firstly, create member id which is mandatory i.e. primary key constraint used.

Creating first name and last name using varchar datatype i.e. variable length

Creating other stuff of the member data e.g.: - address, email id etc.

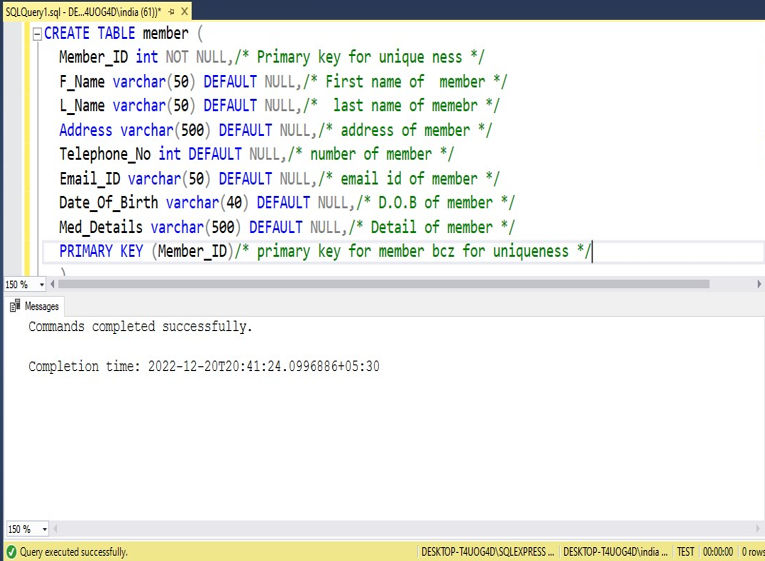
Member id should be unique so author used primary key to identify each record

**Challenges faced:**

The challenges that are faced during the creating the database was with the queries written for adding the constraints for the information mentioned in the statement with respect to the limits over some parameters, other challenges that was faced was with the decision that weather to add some references attributes of one entity into another.

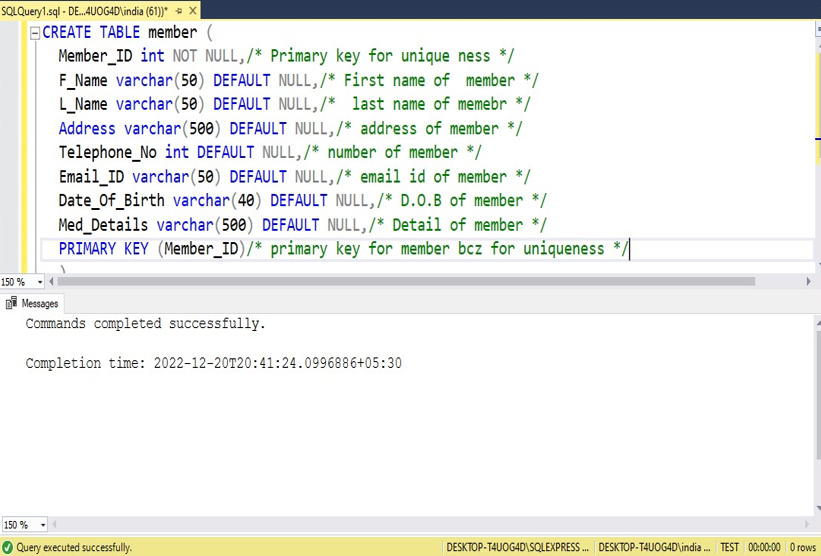
Even if you use lower\_case\_table\_names=2 (which enables MySQL to remember the case used for databases and table names), MySQL does not remember the case used for database names for the function [DATABASE()](https://dev.mysql.com/doc/refman/8.0/en/information-functions.html#function_database) or within the various logs (on case-insensitive systems).

With statement-based binary logging, the source server writes the executed queries to the binary log. This is a very fast, compact, and efficient logging method that works perfectly in most cases. However, it is possible for the data on the source and replica to become different if a query is designed in such a way that the data modification is nondeterministic (generally not a recommended practice, even outside of replication).



**Creating Table for Activity data**

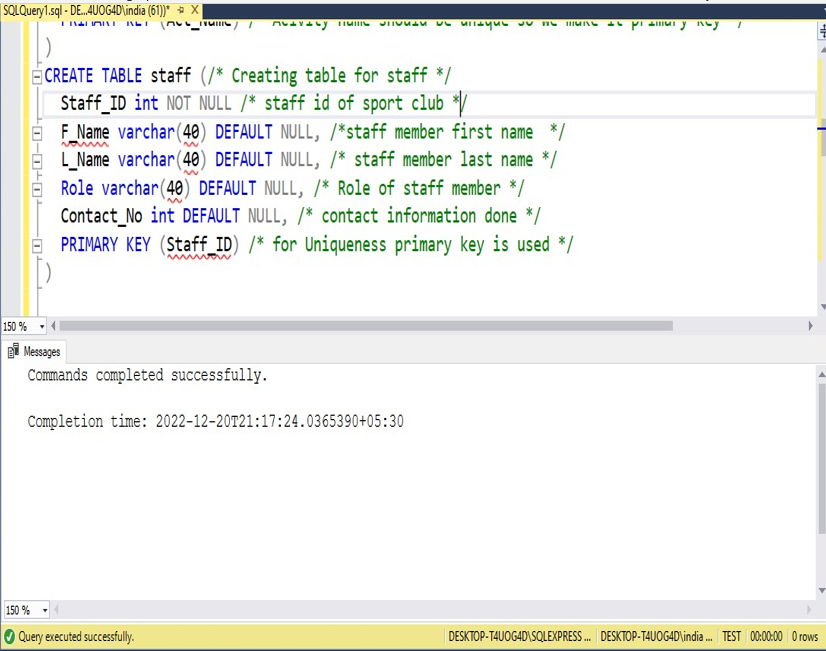
Creating data member for activity table. Number one is activity name that should be unique. After that define other stuff data member with appropriate data type



**Creating Table for Staff data**

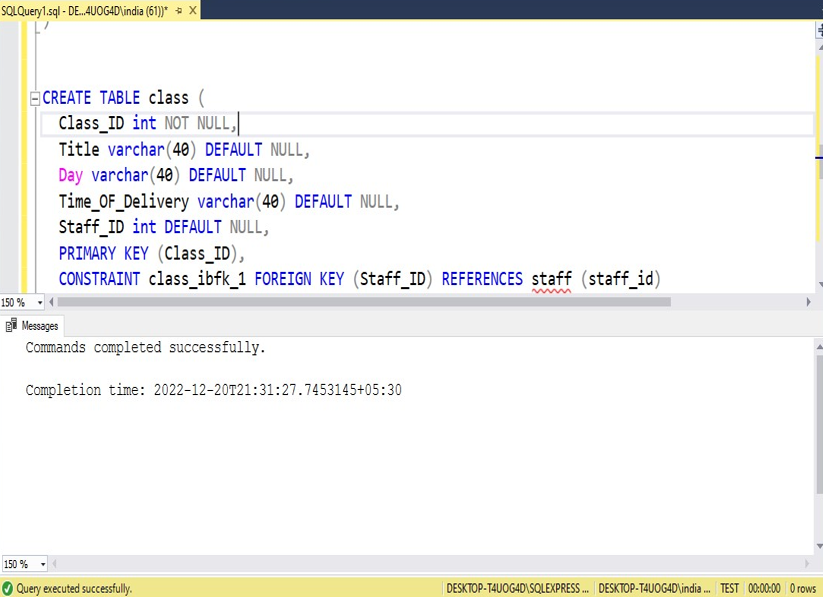
Staff data contain the staff id which is unique and some other stuff data of staff which shown below. Staff data contain first name and last name which has varchar datatype and also contain contact number which has a data type of integer type and we make staff id as primary key to identify each record in a table.

.



**Creating Table for Class data**

Class table contain personal data as well as other table data as a reference foreign key helper.

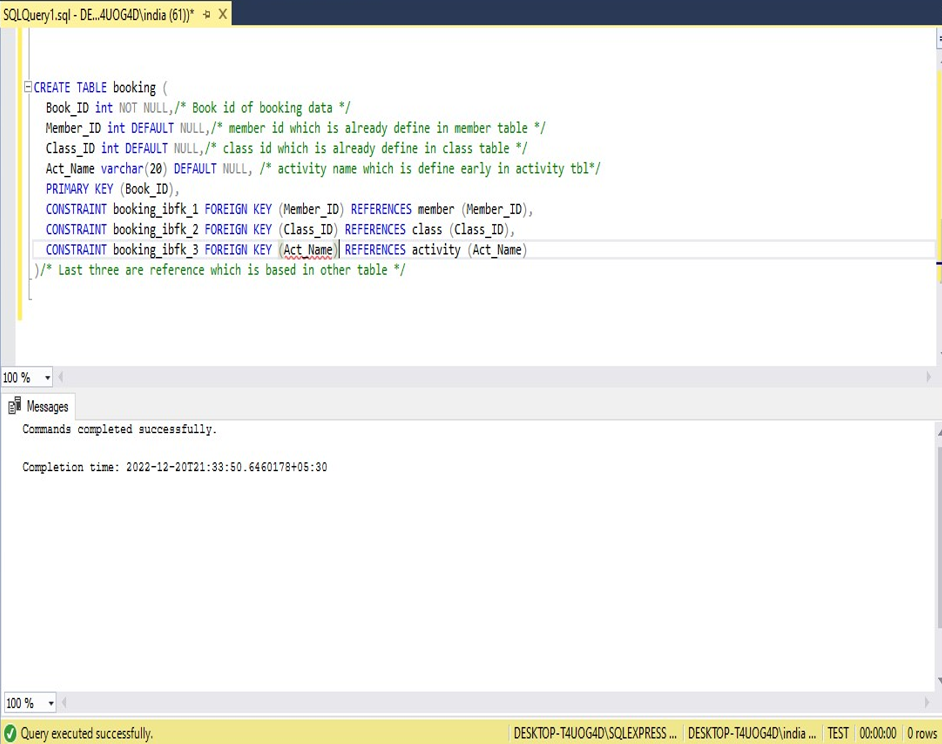


**Creating Table for Booking data**

In Booking table other table data also include which is referencing through various table present in data base. It is done by foreign key.

Booking id should be unique so that identified correctly in every row each.it is done with the help of primary key , Author can define primary key in the starting or may add at the end of all queries.

Voter id , class id and activity name are referenced from other table with the help of foreign key constraint .By the use of foreign key we can reference to other table by mentioning the proper reference and syntax also.

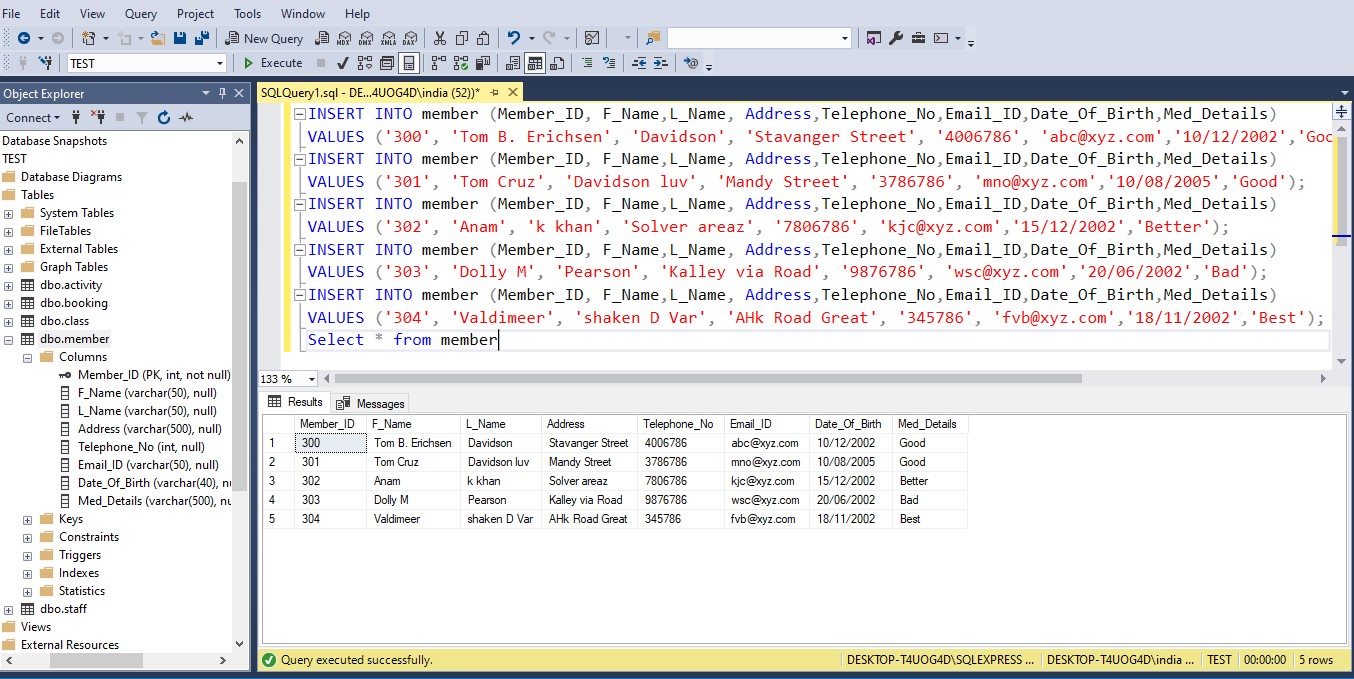


**Working:**

**Adding data and table:**

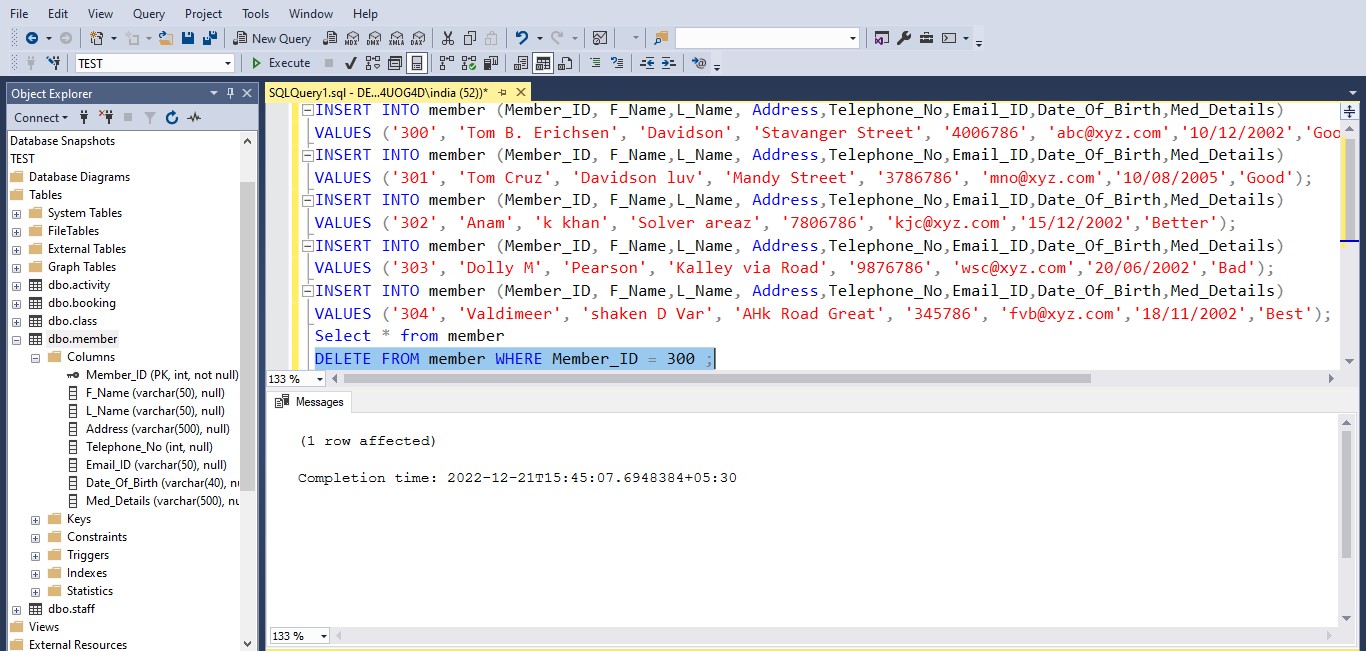
In this we have add some data to check the conditions and validation of the queries.

Author use Insert queries for inserting the data and use select command for displaying the data in the data bar area.



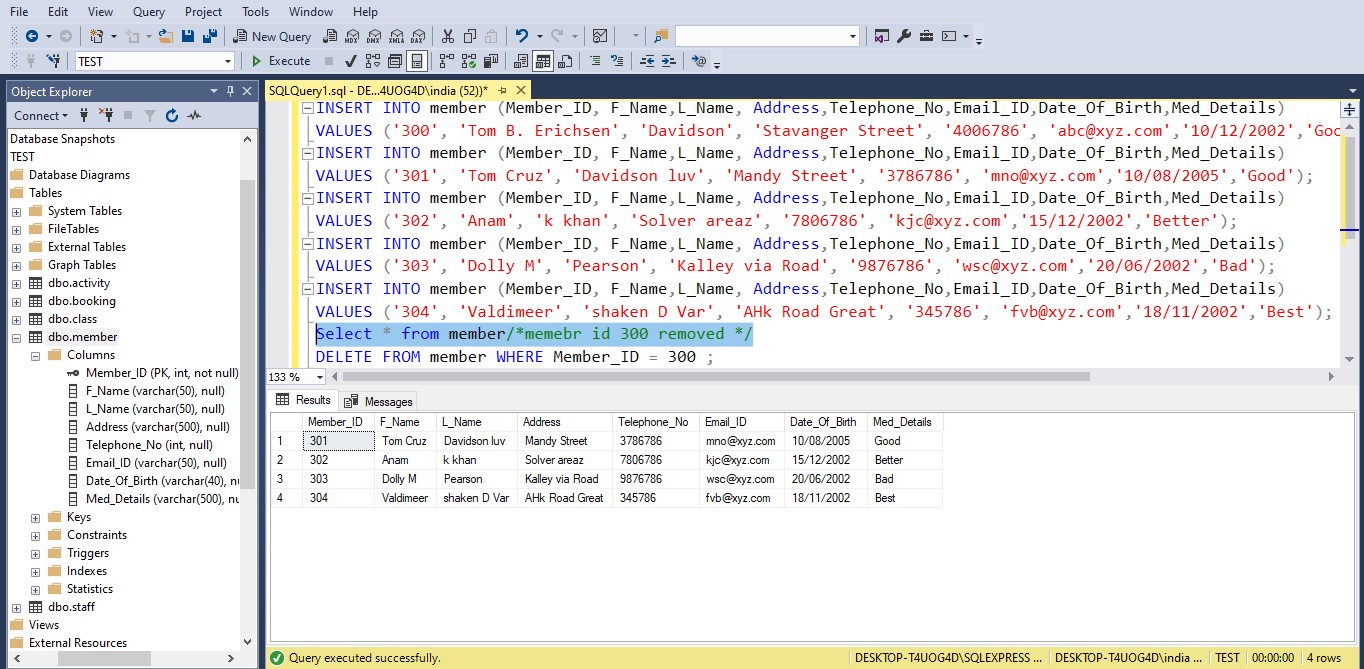
**Remove data:**

With the help of delete command we have to remove the data if needed. In this we have delete one row where id is 300



**Showing Data:**

With the help of select command we can show the data and then also check how many rows are affected. In this one row affected show below in the diagram.



**SPORTS.SQL**

-- MySQL dump 10.13  Distrib 8.0.31, for Win64 (x86\_64)

--

-- Host: localhost    Database: sports

-- ------------------------------------------------------

-- Server version 8.0.31

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `activity`

--

DROP TABLE IF EXISTS `activity`;

/\*!40101 SET @saved\_cs\_client     = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\* created activity Table\*/

CREATE TABLE `activity` (

  `Act\_Name` varchar(25) NOT NULL,

  `Start\_time` varchar(10) DEFAULT NULL,

  `End\_Time` varchar(10) DEFAULT NULL,

  PRIMARY KEY (`Act\_Name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `activity`

--

LOCK TABLES `activity` WRITE;

/\*!40000 ALTER TABLE `activity` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `activity` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `booking`

--

DROP TABLE IF EXISTS `booking`;

/\*!40101 SET @saved\_cs\_client     = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\* Created Table Booking\*/

CREATE TABLE `booking` (

  `Book\_ID` int NOT NULL,

  `Member\_ID` int DEFAULT NULL,

  `Class\_ID` int DEFAULT NULL,

  `Act\_Name` varchar(20) DEFAULT NULL,

  PRIMARY KEY (`Book\_ID`),

  KEY `Member\_ID` (`Member\_ID`),

  KEY `Class\_ID` (`Class\_ID`),

  KEY `Act\_Name` (`Act\_Name`),

  CONSTRAINT `booking\_ibfk\_1` FOREIGN KEY (`Member\_ID`) REFERENCES `member` (`Member\_ID`),

  CONSTRAINT `booking\_ibfk\_2` FOREIGN KEY (`Class\_ID`) REFERENCES `class` (`Class\_ID`),

  CONSTRAINT `booking\_ibfk\_3` FOREIGN KEY (`Act\_Name`) REFERENCES `activity` (`Act\_Name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `booking`

--

LOCK TABLES `booking` WRITE;

/\*!40000 ALTER TABLE `booking` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `booking` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `class`

--

DROP TABLE IF EXISTS `class`;

/\*!40101 SET @saved\_cs\_client     = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*Created Table class\*/

CREATE TABLE `class` (

  `Class\_ID` int NOT NULL,

  `Title` varchar(40) DEFAULT NULL,

  `Day` varchar(40) DEFAULT NULL,

  `Time\_OF\_Delivery` varchar(40) DEFAULT NULL,

  `Staff\_ID` int DEFAULT NULL,

  PRIMARY KEY (`Class\_ID`),

  KEY `Staff\_ID` (`Staff\_ID`),

  CONSTRAINT `class\_ibfk\_1` FOREIGN KEY (`Staff\_ID`) REFERENCES `staff` (`Staff\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `class`

--

LOCK TABLES `class` WRITE;

/\*!40000 ALTER TABLE `class` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `class` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `member`

--

DROP TABLE IF EXISTS `member`;

/\*!40101 SET @saved\_cs\_client     = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*Created Table Mamber\*/

CREATE TABLE `member` (

  `Member\_ID` int NOT NULL,

  `F\_Name` varchar(50) DEFAULT NULL,

  `L\_Name` varchar(50) DEFAULT NULL,

  `Address` varchar(500) DEFAULT NULL,

  `Telephone\_No` int DEFAULT NULL,

  `Email\_ID` varchar(50) DEFAULT NULL,

  `Date\_Of\_Birth` varchar(40) DEFAULT NULL,

  `Med\_Details` varchar(500) DEFAULT NULL,

  PRIMARY KEY (`Member\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `member`

--

LOCK TABLES `member` WRITE;

/\*!40000 ALTER TABLE `member` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `member` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `staff`

--

DROP TABLE IF EXISTS `staff`;

/\*!40101 SET @saved\_cs\_client     = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*Created Table Staff\*/

CREATE TABLE `staff` (

  `Staff\_ID` int NOT NULL,

  `F\_Name` varchar(40) DEFAULT NULL,

  `L\_Name` varchar(40) DEFAULT NULL,

  `Role` varchar(40) DEFAULT NULL,

  `Contact\_No` int DEFAULT NULL,

  PRIMARY KEY (`Staff\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `staff`

--

LOCK TABLES `staff` WRITE;

/\*!40000 ALTER TABLE `staff` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `staff` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2022-12-13 17:09:04

**Testing Report:**

The Database is one of the inevitable parts of a Software Application.

It does not matter whether it is a web, desktop or mobile, client-server, peer-to-peer, enterprise, or individual business; the Database is required everywhere at the backend.

Similarly, whether it is Healthcare, Finance, Leasing, Retail, Mailing application, or controlling a spaceship; a Database is always in action behind the scene.

As the complexity of application increases, the need for a stronger and secure Database emerges. In the same way, for the applications with a high frequency of transactions (e.g; Banking or Finance application), the necessity of a fully-featured DB Tool is coupled.

Nowadays, we have big data that is large and complex that the traditional databases can’t handle them.

There are several database tools are available in the market e.g; MS-Access, MS SQL Server, SQL Server, Oracle, Oracle Financial, MySQL, PostgreSQL etc. These tools vary in cost, robustness, features, and security. Each of these has its own benefits and drawbacks.

**System Tech Design Report:**

Replication Schemes:

There are three ways of doing replication:

**Synchronous replication**: Changes are propagated to all the database nodes before considering the transaction successful. If any of the operations fails, the entire transaction fails and the changes are rolled back.

**Asynchronous replication**: Changes are propagated asynchronously after the transaction has been considered successful.

**Semi-synchronous replication**: The replication is partially synchronous. A transaction is considered successful after getting replicated on x number of nodes. The node does not wait for acknowledgment from other nodes.

**Types of replication architectures**:

There are two major architectures based on database replication:

**Master-slave architecture**:

In a master-slave architecture, the master node serves the 'read and 'write' requests. The master node replicates the changes to the slave nodes. The slave nodes only serve 'read' requests.

If the master dies, one of the slave nodes is promoted as the master, or a new master is provisioned.

A standby master can also be considered. It is a replica of the active master. It becomes the active master if the active master dies.

**Advantage:**

The 'read redundancy' (slaves) distributes the 'read' traffic. Adding 'read redundancy' can scale the system as the traffic increases.

The system can continue to serve 'read' requests even when the master node dies. There is no single point of failure for 'read' requests.

**Disadvantages:**

Replication adds overhead. It might add latency if we choose synchronous replication. It might result in inconsistency if we choose asynchronous replication.

Consider that the master dies and the replication has not been completed. In this case, there might be some data loss as the newly promoted master might not have the latest changes.

The load on the master node might still be high if the number of 'writes' is high.

Replication adds a lot of complexity to the system.

**Multi-master architecture**:

In a multi-master architecture, all the nodes act as the master and can serve both 'reads' and 'writes'.

If any of the master nodes die, the system can operate as usual.

**Advantages:**

The 'read' and 'write' requests are distributed. We can scale the system by adding redundancy if traffic increases.

The system can continue to serve requests even when any of the nodes die. There is no single point of failure.

There is no need to select a new master when any of the master nodes die.

**Disadvantages:**

Replication adds overhead. It might add latency if we choose synchronous replication. It might result in inconsistency if we choose asynchronous replication.

Conflict prevention (in case of sync replication) or conflict resolution (in case of async replication) might be a big overhead.

Consider that the master dies and the replication has not been completed. In this case, there might be some data loss as the other master nodes might not have those changes.

Replication adds a lot of complexity to the system.

# Summary:

**Document databases** – A kind of non-relational database designed for storing and querying data as JSON-like documents. Document databases ease a developer’s effort for storing and querying data in a database by using the same document-model format used in the application code. **ER model databases**– A high-level conceptual data model diagram, ER or entity-relationship modeling helps in systematically analyzing data requirements for producing a well-designed database. **Graph databases** – A type of NoSQL database, a Graph database replace tables with graphs having vertices and edges. While the vertices represent entities, such as a person or place, the edges represent the relationship amongst two nodes. **Hierarchical databases** – Follows a data model in which the data is organized into a tree-like structure. The data is stored as records, and different records are connected via links. Each record contains a collection of fields, whereas each field has only one value. **NoSQL databases** – An alternative to the traditional relational database, a NoSQL database is immensely useful when working with giant sets of distributed data. Unlike SQL databases that are table-based, NoSQL databases are document-based, key-value pairs, graph databases, or wide-column stores. **Network databases** – A form of database model in which it is possible to link several member records or files to multiple owner files and vice-versa is a network database. The model is like an upside-down tree where each member information is the branch linked to the owner, represented by the bottom of the tree. **Object-oriented databases** – A niche in the mammoth RDBMS industry, an object-oriented database model is one in which objects represent the information. This one is much less popular than other types of database management systems.

**Other Point Remember**:-

### **Improved efficiency**

Database management systems are essential for businesses because they offer an efficient way of handling large amounts and multiple types of data. The ability to access data efficiently allows companies to make informed decisions quicker.

### **Versatility**

Database management systems are incredibly versatile. They can be accessed on computers, tablets, and even mobile devices.

### **Allowing categorization and structuring of available data**

Database management systems allow organizations and individuals to categorize and structure available data. For instance, a database for a university can store Sport club information (e.g., names, contact number, activity name etc.).

### **Creating an organized working environment**

A database management system offers a way to create a smooth, more organized working environment. A database query language, such as SQL, provides an easy way to access, update, as well as process data stored in databases.

Most database management systems come prepackaged with programs that have Microsoft SQL Server and MySQL queries for enabling 3rd-party programs to access data.

**Newer and better ways to manage data**

A robust database management system not only allows authorized users to add new data to the database while also updating the present data and deleting any data that might have become obsolete. Like the world, the database management system is evolving.

There is a constant addition of new and better ways of doing all DBMS-related things. [Machine learning has allowed DBMSs to generate suggestions](https://aws.amazon.com/blogs/machine-learning/tuning-your-dbms-automatically-with-machine-learning/) that can help a decision-maker or admin better manage the database.

The project was based on the application of SQL and its use for the designing and implementing the database for sport management club. The project start with the designing the ER diagram for the project. To understand the process we needed the activity diagram and to understand the functionality of each member in the diagram we used use-case diagram.

Project then continued with the implementation of database in relational database model, written a query for making tables for member, class, staff, activity and booking with different attributes for each entity.

The working part of the project included the add, update and delete queries. This queries worked perfectly to satisfy the statements mentioned.