## Q1- A comparison of different database systems

Oracle, MySQL, PostgreSQL, Microsoft SQL, MongoDB, Redis, MariaDB, Firebase and Elasticsearch are among the most popular Database management systems at the moment.

* **Relational / SQL based – Organizing data into tables that are related to one another, built over SQL. These are more widely known than NoSQL options.**

1. *Oracle :* Oracle RDBMS offers extensive support and documentation, and charges a hefty fee of $17500 per unit just for the standard edition, the Enterprise edition running for over $47000 per unit. It does offer one of the larger processing capacity but that also makes it quite resource consuming.
2. *MySQL :* The most popular relational database system, this one is also open-source, well used to be till it was acquired by Oracle. Being free as well as beginner friendly, much easier to use than Oracle, is the reason for its undying popularity. But it does not work well with systems which depend on scalability. Also, the partial open-source nature brings its share of problems with some implementations not working.
3. *PostgreSQL :* Popular due to the user-defined objects combined with tables. It is similar to MySQL in many ways, it is also better than it, especially regarding data scalability and support for custom (relatively new) data types like JSON or XML. It is also open-source but the documentation is not on the level of Oracle, lacks consistency and completeness.
4. *MSSQL* *:* Microsoft SQL Server is up there in popularity, partly due to the variety of the options available. It is a completely commercial tool, and this is the only point on which it loses to PostgreSQl or MySQL, but the rich documentation and the extensive community assistance is only rivaled by Oracle. The integration with Microsoft Azure is the biggest advantage it holds over other database systems.
5. *MariaDB :* This is an open source fork of MySQL with some commercial support available. Encryption is the keyword here. While the security offered is better than MySQL or PostgreSQL, it is leagues above them in terms of performance. It also supports both SQL and NoSQL data. But it is relatively small compared to the other entries and the community support is lacking.

* **Non-relational / NoSQL based – Handles complex applications. Databases take any form desirable, no strict format like SQL.**

1. *MongoDB :* Open source with commercial support available, similar to MariaDB can be used for structured as well as unstructured data, but it has a better community support than MariaDB. It is also excellent with Horizontal scalability and data access, storage, input and retrieval is really simple. It is however on the heavier side regarding memory consumption. Also the insecurity with the community version is a heavy contrast to MariaDB’s default encryption.
2. *Redis :* Open-source NoSQL based system which uses key-value pairs instead of documents. It processes data rapidly and works well with massive data processing. It can also be used as a cache. This makes it somewhat prone to crashing, especially compared to stable options like MySQL forks. Also there is no support for query languages.
3. *Firebase :* Owned by Google, it works as expected; handling large amounts of data in real time with ease. Its beginner friendly nature makes it stand out compared to MongoDB or especially Radis. It has arguably the best documentation available, even better than Microsoft’s and the size of its community is decent. It is however lagging behind in terms of the limited querying capabilities and the limited data migration.
4. *Elasticsearch :* A NoSQL document based management system with a very robust scalable architecture, it is as fast as any other NoSQL based systems, obviously a lot faster than any SQL based system. It does handle complex queries really efficiently unlike Google’s Firebase due to a full-text search engine at its heart. The data processing is fast as expected, it does however lack multi-language support since it only works with JSON.

## **Q2- History of database systems**

* In early 1960s, Charles Bachman developed the Integrated Data Store(IDS) based over the network model, for which he recieved a Turing Award. He founded the Database Task Group within CODASYL. Their approach was called the CODASYL approach.
* In 1966, IBM launched thier own Information Management System (IMS) written on System360. It was similar to CODASYL with a different method of data navigation.
* 1970s saw the introduction of relational DBMS, in a paper published by an IBM employee Edgar Codd. Codd didn’t like CODASYL data navigation approach and used mathematical terms for his model: relations, tuples and domains. Many of the databases we use today are relational.
* IBM launched SQL DBMS in late 1980s, the work on which started in early 1970s under the project “System R”. Larry Ellison’s Oracle Database was also based on IBM’s papers on System R, and their Oracle Version 2 beat IBM to the market in 1979.
* There was a rush of desktop computing in 1980s, with spreadsheet softwares like Lotus and the launch of dBASE, a management system which handled light data for the user under a layer of abstraction. dBASE was one of the top selling software titles of that time.
* Object Oriented Programming saw a rise in 1990s, with it the object-relational mappings (IRM) integration with SQL to handle data as objects. This alllowed for relations between objects and their attributes, further improving data navigation in the system.
* 2000s further improved on the relational model, and also saw the introduction of NoSQL, NewSQL and the use of XML for database management. NoSQL databases are really fast, require no structure or pattern in the data. They are also scalable. With scalability being a sort of issue with SQL, NewSQL aimed to provide a similar speed to NoSQL for online transactions with a base maintained in SQL.

## Q3- SQL and some basic queries

SQL stands for **Structured Query Language**, commands written with it are used to communicate with a relational database to perform storage and navigation tasks. Initially created in 1970s, it came into prominence with the release of IBM SQL DBMS and Oracle’s DBMS.

Uses of SQL include creation and management of tables for storing data, navigating these tables in search of data, deleting or adding rows and columns to existing databases and transaction processing.

The most popular databases are still relational, and the popular relational database management systems are often based on SQL. Microsoft SQL, Oracle, MySQL, MariaDB and PostgreSQL are among the top database systems at the moment, all of which implement SQL.

Some basic commands for SQL –

* Use **CREATE TABLE t**o create the table with desired columns and with their data types in mind using the syntax :

**CREATE TABLE <table\_name> (<col\_name> <data\_type> (element\_size), <col\_name> <data\_type> (element\_size) )**;

* **ALTER TABLE** is used to change the structure of a table with the syntax :

**ALTER TABLE <table\_name>**

There is no semicolon after this statement since ALTER won’t do anything to the table by itself.

* Use ADD to add a column to the database table after calling in the table using ALTER :

**ALTER TABLE <table\_name>**

**ADD column\_name datatype;**

* A **CHECK** constraint can be added to some column of the table:

Example : **CREATE TABLE Movies ( Score number(10), CHECK (Score > 0));**

This will make sure any score added to the table is greater than 0.

* **SELECT** query will show the data stored in some table, with calling some columns:

**SELECT column\_name FROM table;**

Use \* for column\_name to get all the columns and their stored information.

* **WHERE** can be used to limit the results obtained from SELECT :

**<table\_name> <column\_name> WHERE (column BETWEEN A AND B)**

**Example – STUDENT Name, RollNumber WHERE (RollNumber BETWEEN 1 AND 28 OR RollNumber = 40);**

* **UPDATE** is used to update a record in the table:

**UPDATE table\_name**

**SET col1 = val1, col2 = val2,**

**WHERE condition;**

## Q4- Simple DBMS functioning using a virtual lab

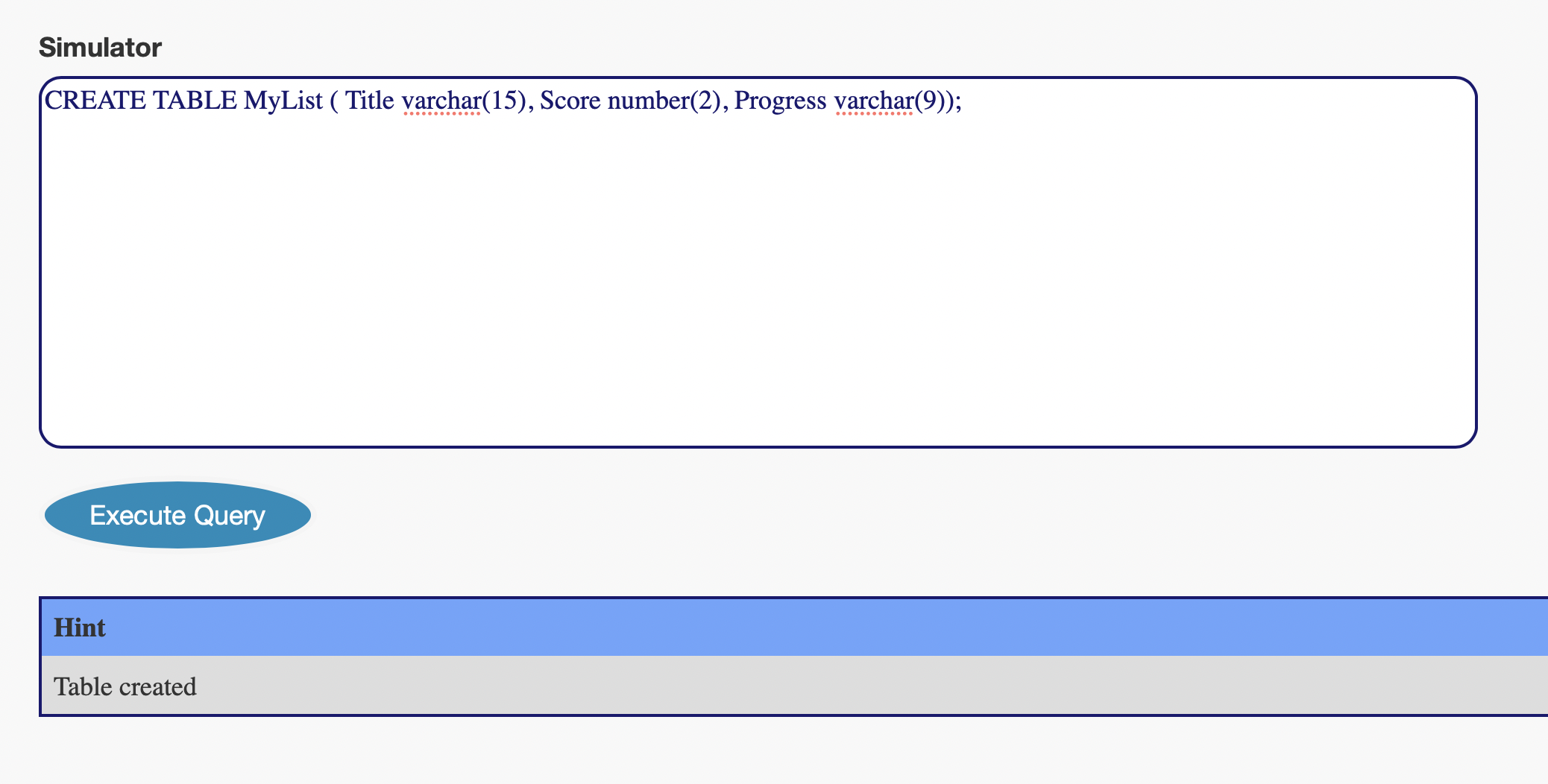
IIT Bombay hosts a virtual Database Lab, in collaboration with the Department of Computer Science of Gujarat University. One can access the site from the link : <http://vlabs.iitb.ac.in/vlabs-dev/labs/dblab/index.php>

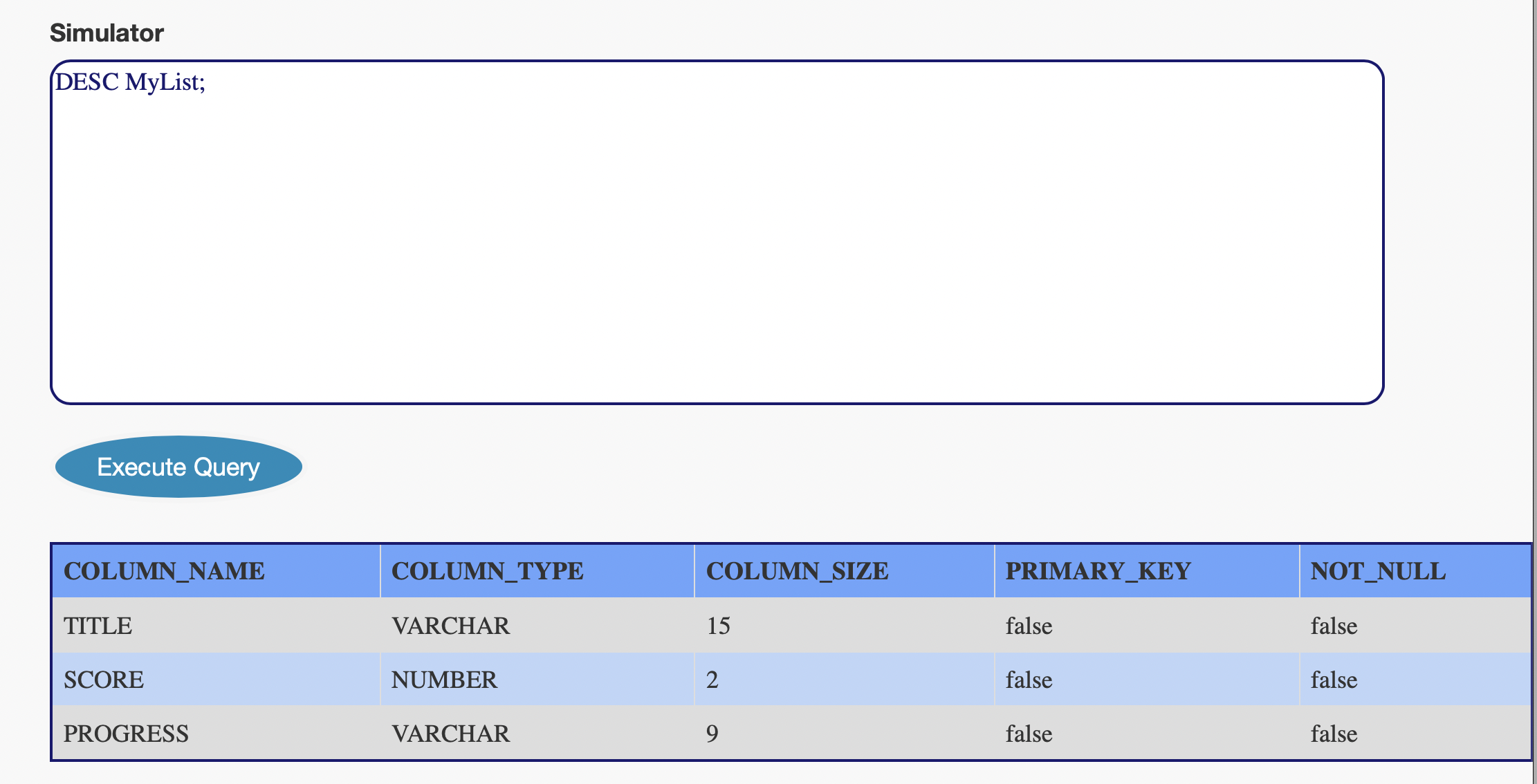
**AIM –** To get familiar with some of the basic SQL queries.

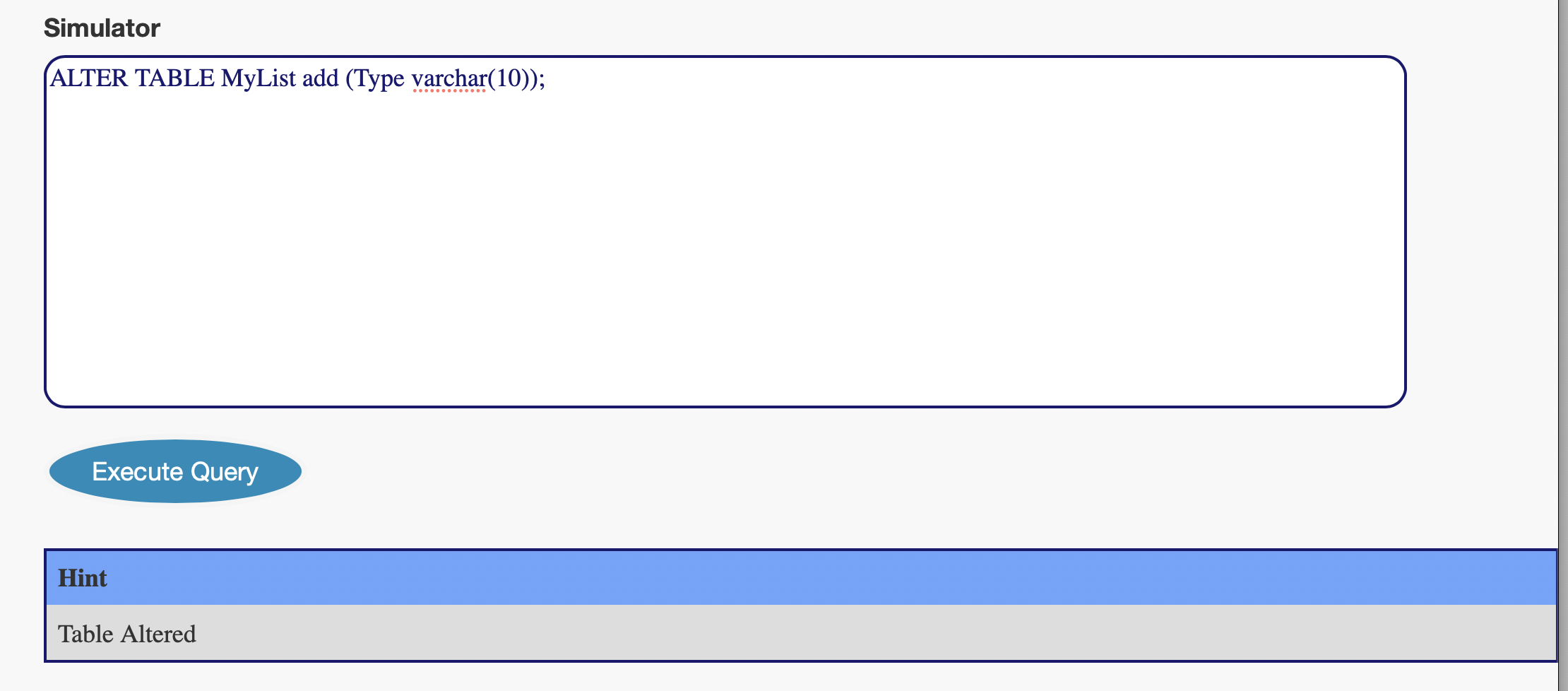
**WORKING –**

* Creating a simple table for storing the information of movies and shows watched over a weekend

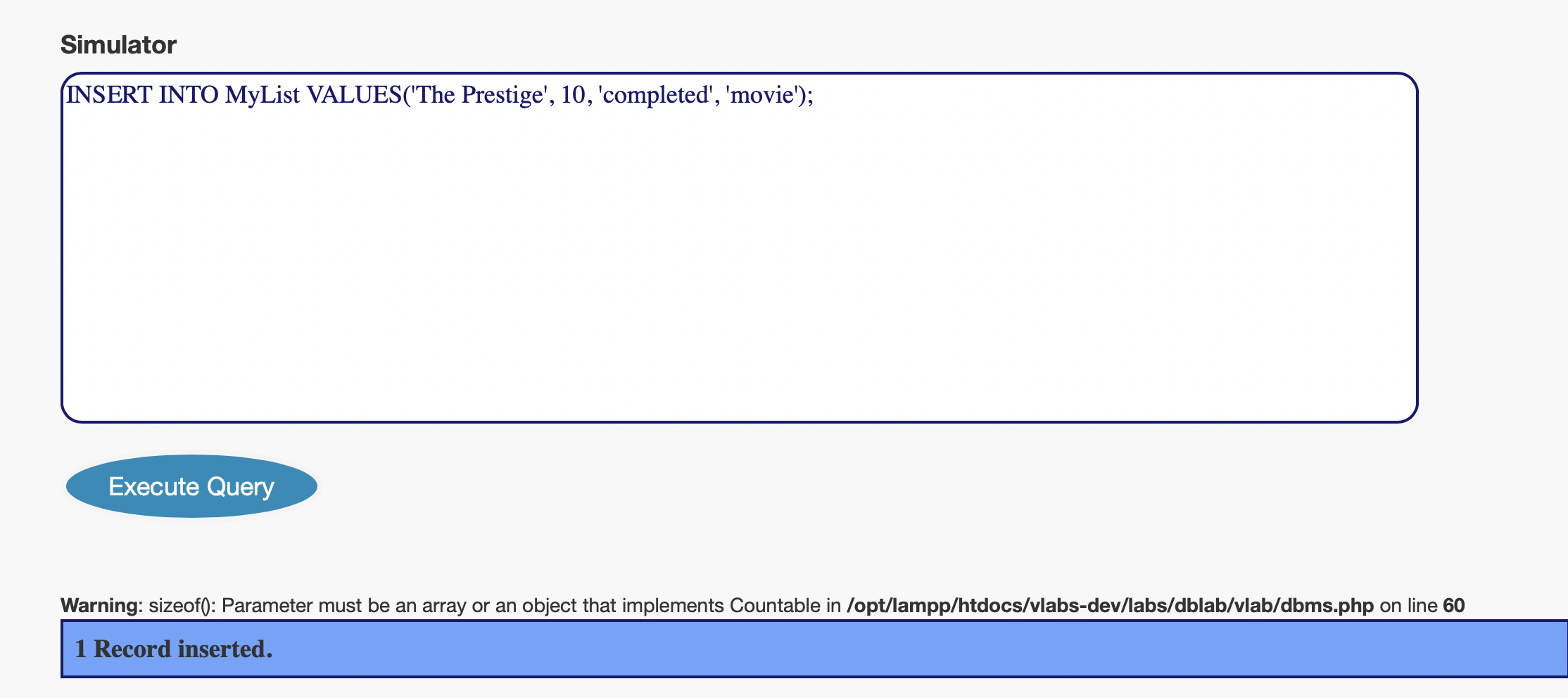
**CREATE TABLE MyList ( Title varchar(15), Score number(2), Progress varchar(9));**



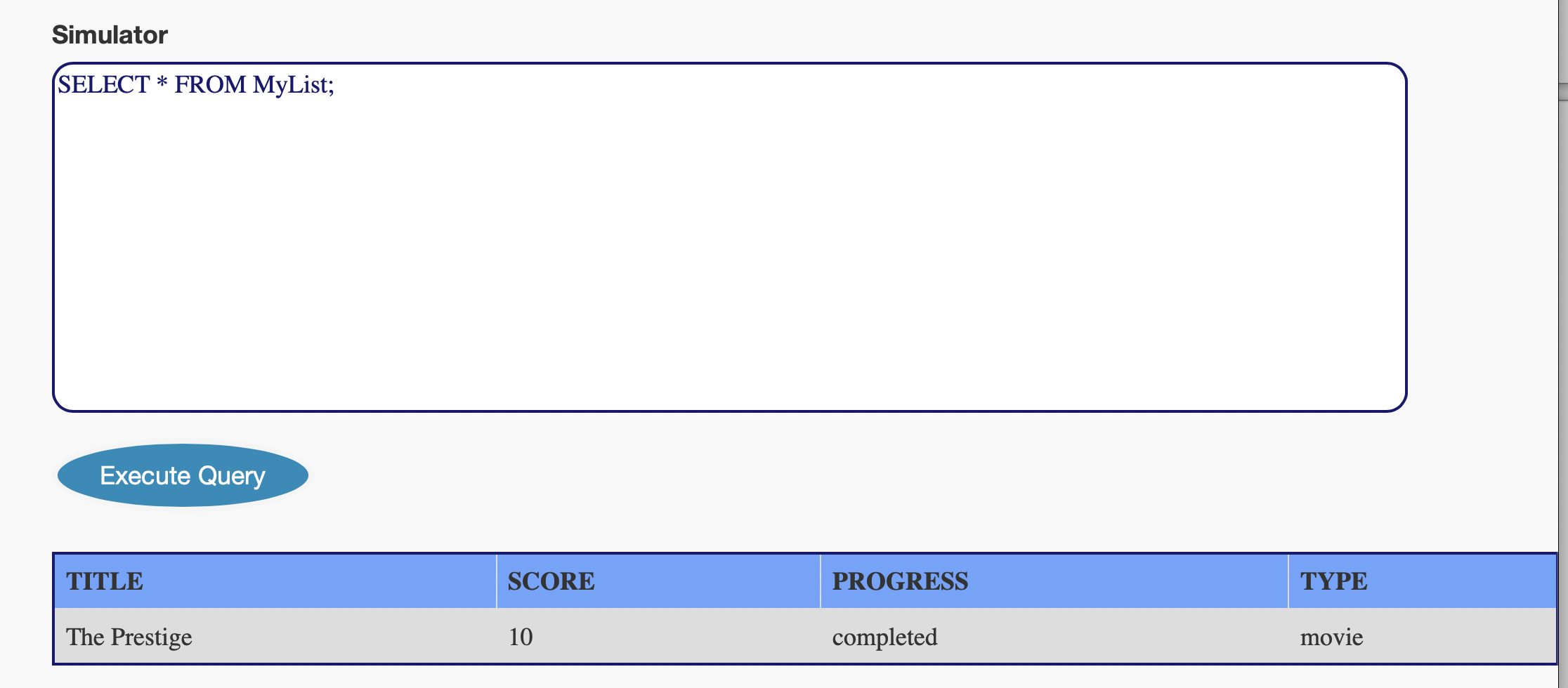
* *DESC* query is used to check the table created or any other table and its columns.  
    
  **DESC <table\_name>;**
* Using *ADD* to add a column to the table after using *ALTER* command.

**ALTER TABLE <table> ADD(col\_name data\_type(size));**

* Use *INSERT INTO* to insert values into the table



* Use *SELECT <col>* to check values of a column, \* to check all the values of all the columns.



## Q5- A sample Database for hospital management

**Patient database**

|  |  |  |
| --- | --- | --- |
| COLUMN\_NAME | COLUMN\_TYPE | SIZE |
| **Name** | varchar | 20 |
| **Age** | number | 3 |
| **Room Allocated** | number | 4 |
| **Stay duration** | date |  |

**Medicine database**

|  |  |  |
| --- | --- | --- |
| COLUMN\_NAME | COLUMN\_TYPE | SIZE |
| **Name** | varchar | 30 |
| **Medication ID** | varchar | 50 |
| **Stock remaining** | number | 10 |

**Employee database**

|  |  |  |
| --- | --- | --- |
| COLUMN\_NAME | COLUMN\_TYPE | SIZE |
| **Name** | varchar | 15 |
| **Employee id** | number | 10 |
| **Position** | varchar | 10 |
| **Address** | varchar | 50 |
| **Salary** | number | 7 |