What is MySQL

MySQL is a fast, easy to use relational database. It is currently the most popular open-source database. It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications.
Relational Database Management System (RDBMS): MySQL is a relational database management system.

Client/ Server Architecture: MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
Allows roll-back: MySQL allows transactions to be rolled back, commit and crash recovery.

MySQL is used for many small and big businesses. It is developed, marketed and supported by MySQL AB, a Swedish company. It is written in C and C++.
A database management system (DBMS) enables users to create, read, update and delete data in a database. It lets end users use data while managing data integrity. It stores data that can be shared by multiple users in a controlled manner concurrently.

A relational database management system (RDBMS) is based on the relational model. This model uses relationship between tables using primary keys, foreign keys and indexes.

RDBMS uses constraints of primary and foreign keys to establish relationships between rows of data in different database tables. It uses the concept of normalization to eliminate the need to redundantly store related data in multiple tables.

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

SQL stands for Structured Query Language, is a language to communicate with the Database. It performs tasks such as retrieval, updation, insertion and deletion of data from a database.

SQL is a standard language for accessing and manipulating databases.

SQL can execute queries against a database
SQL can retrieve data from a database
SQL can insert records in a database
SQL can update records in a database
SQL can delete records from a database
SQL can create new databases
SQL can create new tables in a database
SQL can create stored procedures in a database
SQL can create views in a database
SQL can set permissions on tables, procedures, and views

Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.

Some of The Most Important SQL Commands
SELECT - extracts data from a database
UPDATE - updates data in a database
DELETE - deletes data from a database
INSERT INTO - inserts new data into a database
CREATE DATABASE - creates a new database
ALTER DATABASE - modifies a database
CREATE TABLE - creates a new table
ALTER TABLE - modifies a table
DROP TABLE - deletes a table
CREATE INDEX - creates an index (search key)
DROP INDEX - deletes an index

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

A primary key is a unique identifier that uniquely identify a record in a database table. It has implicit NOT NULL constraint (means primary key values can't be NULL).

A Unique key ensures rows are unique within the database. It provides uniqueness of the column on which it is defined.

Difference between a Primary Key and a Unique Key.
Answer:

A primary key job is to uniquely identify a row within a table while a Unique key ensures additional unique conditions on column(s).

A primary key enforces entity integrity whereas Unique key enforces unique data. There may be many unique key constraints for one table, but only one PRIMARY KEY constraint for one table.

Primary key doesn't allow NULLs, but unique key allows one NULL only.

Primary key creates a clustered index on the column by default, whereas unique creates a non-clustered index by default.

A foreign key is a column or group of columns that establishes a link between the data in two tables. A foreign key ensures referential integrity of the data.

What are the different types of SQL statements?
Answer:

1. DDL - Data Definition Language
DDL defines the structure that holds the data. It creates and modifies the structure of database objects in database. For example, Create, Alter, Drop and

Truncate table.

2. DML - Data Manipulation Language
It is used to retrieve, modify, delete, insert data in database. Typical operations are Insert, Delete, Update and retrieving the data from the table.

3. DCL - Data Control Language
DCL controls the visibility of data and enforces database security in a multiple user database environment. It includes operations like granting database access, setting privileges to create tables etc. Two types of DCL commands are GRANT and REVOKE.

4. Transaction Control Language (TCL)
TCL commands are used to manage transactions in database. e.g. COMMIT and ROLLBACK.

MySQL SELECT Database
SELECT Database is used in MySQL to select a particular database to work with. This query is used when multiple databases are available with MySQL Server.

You can use SQL command USE to select a particular database.

MySQL Drop Database
You can drop/delete/remove a MySQL database easily with the MySQL command. You should be careful while deleting any database because you will lose your all the data available in your database.

MySQL CREATE TABLE
The MySQL CREATE TABLE command is used to create a new table into the database. A table creation command requires three things:

Name of the table
Names of fields
Definitions for each field
Syntax:

Following is a generic syntax for creating a MySQL table in the database.

```
CREATE TABLE table_name (column_name column_type...);
```

See the created table:

Use the following command to see the table already created:

```
SHOW tables;
```

See the table structure:

Use the following command to see the table already created:

```
DESCRIBE cus_tbl;
```

MySQL ALTER Table
MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation.

1) ADD a column in the table
Syntax:

```
ALTER TABLE table_name
ADD new_column_name column_definition
[ FIRST | AFTER column_name ];
```

Parameters
table_name: It specifies the name of the table that you want to modify.

new_column_name: It specifies the name of the new column that you want to add to the table.

column_definition: It specifies the data type and definition of the column (NULL or NOT NULL, etc).

FIRST | AFTER column_name: It is optional. It tells MySQL where in the table to create the column. If this parameter is not specified, the new column will be added to the end of the table.

Example:

In this example, we add a new column "cus_age" in the existing table "cus_tbl".

Use the following query to do this:

```
ALTER TABLE cus_tbl
ADD cus_age varchar(40) NOT NULL;
```

MODIFY column in the table
The MODIFY command is used to change the column definition of the table.

Syntax:

```
ALTER TABLE table_name
MODIFY column_name column_definition
[ FIRST | AFTER column_name ];
```

Example:

In this example, we modify the column cus_surname to be a data type of varchar(50) and force the column to allow NULL values.

```
ALTER TABLE cus_tbl
MODIFY cus_surname varchar(50) NULL;
```

DROP column in table
Syntax:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Let's take an example to drop the column name "cus_address" from the table "cus_tbl".

Use the following query to do this:

```
ALTER TABLE cus_tbl
DROP COLUMN cus_address;
```

RENAME column in table
Syntax:

```
ALTER TABLE table_name
CHANGE COLUMN old_name new_name
column_definition
[ FIRST | AFTER column_name ]
```

Example:

In this example, we will change the column name "cus_surname" to "cus_title".

Use the following query to do this:

```
 ALTER TABLE  cus_tbl
CHANGE COLUMN cus_surname cus_title
varchar(20) NOT NULL;
```

RENAME table
Syntax:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

Example:

In this example, the table name cus_tbl is renamed as cus_table.

```
ALTER TABLE cus_tbl
RENAME TO cus_table;
```

MySQL TRUNCATE Table
MYSQL TRUNCATE statement removes the complete data without removing its structure.

The TRUNCATE TABLE statement is used when you want to delete the complete data from a table without removing the table structure.

Syntax:

```
TRUNCATE TABLE  table_name;
```

MySQL DROP Table
MYSQL DROP table statement removes the complete data with structure.

Syntax:

```
DROP TABLE  table_name;
```

MySQL TRUNCATE Table vs DROP Table
You can also use DROP TABLE command to delete complete table but it will remove complete table data and structure both. You need to re-create the table again if you have to store some data. But in the case of TRUNCATE TABLE, it removes only table data not structure. You don't need to re-create the table again because the table structure already exists.

The SQL DELETE Statement
The DELETE statement is used to delete existing records in a table.

DELETE Syntax
```
DELETE FROM table_name WHERE condition;
```

MySQL View
In MySQL, View is a virtual table created by a query by joining one or more tables.

MySQL View
In MySQL, View is a virtual table created by a query by joining one or more tables.

MySQL Create VIEW
A VIEW is created by SELECT statements. SELECT statements are used to take data from the source table to make a VIEW.

Syntax:

```
CREATE [OR REPLACE] VIEW view_name AS
SELECT columns
FROM tables
[WHERE conditions];
```

ex. Create view Amit AS Select FirtsName,LastName From Student;

Explanation : above mention command(query) will create view Amit(virtual table) with tow columns FirstName and LastName

To see the view query is Select * from Amit;

UPDATE
Update is the DML query if we want to update any record is table we use UPDATE Statement to modify existing data in table

UPDATE Syntax
```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Example:
```
UPDATE Employees SET LastName="Amitsing" ,FirstName="Chaudhari" Where EmployeeID='1';
```

UPDATE Multiple Records
It is the WHERE clause that determines how many records that will be updated.

```
Update OrderDetails Set  ProductId='15' where OrderID='10267';
Select * from OrderDetails  where OrderID='10267';
```

MySQL Update VIEW
In MYSQL, the ALTER VIEW statement is used to modify or update the already created VIEW without dropping it.

Syntax:

```
ALTER VIEW view_name AS
SELECT columns
FROM table
WHERE conditions;
```
Example:

The following example will alter the already created VIEW name "trainer" by adding a new column.

```
ALTER VIEW trainer AS
SELECT course_name, course_trainer, course_id
FROM courses;
```

MySQL Drop VIEW
You can drop the VIEW by using the DROP VIEW statement.

Syntax:

```
DROP VIEW [IF EXISTS] view_name;
```
Parameters:
view_name: It specifies the name of the VIEW that you want to drop.

IF EXISTS: It is optional. If you do not specify this clause and the VIEW doesn't exist, the DROP VIEW statement will return an error.

Example:

```
DROP VIEW trainer;
```

MySQL Queries

A list of commonly used MySQL queries to create database, use database, create table, insert record, update record, delete record, select record, truncate table and drop table are given below.

1) MySQL Create Database

MySQL create database is used to create database. For example

```
create database db1;
```

```
CREATE DATABASE database_name;
```

Example:

Let's take an example to create a database name "employees"

```
CREATE DATABASE employees;
```

You can check the created database by the following query:

```
SHOW DATABASES;
```

All the database names, table names and table fields name are case sensitive. You must have to use proper names while giving any SQL command.

MySQL SELECT Database

SELECT Database is used in MySQL to select a particular database to work with. This query is used when multiple databases are available with MySQL Server.

You can use SQL command USE to select a particular database.

Syntax:

```
USE database_name;
```

Example:

Let's take an example to use a database name "customers".

```
USE customers;
```

MySQL Drop Database

You can drop/delete/remove a MySQL database easily with the MySQL command. You should be careful while deleting any database because you will lose your all the data available in your database.

Syntax:

```
DROP DATABASE database_name;
```

Example:

Let's take an example to drop a database name "employees"

```
DROP DATABASE employees;
```

MySQL CREATE TABLE
The MySQL CREATE TABLE command is used to create a new table into the database. A table creation command requires three things:

Name of the table
Names of fields
Definitions for each field

Here, we will create a table named "cus_tbl" in the database "customers".

```
CREATE TABLE cus_tbl(
   cus_id INT NOT NULL AUTO_INCREMENT,
   cus_firstname VARCHAR(100) NOT NULL,
   cus_surname VARCHAR(100) NOT NULL,
   PRIMARY KEY ( cus_id )
);
```

Here, NOT NULL is a field attribute and it is used because we don't want this field to be NULL. If you will try to create a record with NULL value, then MySQL will raise an error.
The field attribute AUTO_INCREMENT specifies MySQL to go ahead and add the next available number to the id field.PRIMARY KEY is used to define a column as primary key. You can use multiple columns separated by comma to define a primary key.

See the table structure:

Use the following command to see the table already created:

```
DESCRIBE cus_tbl;
```

MySQL ALTER Table
MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation.

1) ADD a column in the table
Syntax:

```
ALTER TABLE table_name
ADD new_column_name column_definition
[ FIRST | AFTER column_name ];
```

Parameters
table_name: It specifies the name of the table that you want to modify.

new_column_name: It specifies the name of the new column that you want to add to the table.

column_definition: It specifies the data type and definition of the column (NULL or NOT NULL, etc).

FIRST | AFTER column_name: It is optional. It tells MySQL where in the table to create the column. If this parameter is not specified, the new column will be added to the end of the table.

Example:

In this example, we add a new column "cus_age" in the existing table "cus_tbl".

Use the following query to do this:

```
ALTER TABLE cus_tbl
ADD cus_age varchar(40) NOT NULL;
```

Add multiple columns in the table
Syntax:

```
 ALTER TABLE table_name
 ADD new_column_name column_definition
 [ FIRST | AFTER column_name ],
ADD new_column_name column_definition
[ FIRST | AFTER column_name ],
  ...
;
```

Example:

In this example, we add two new columns "cus_address", and cus_salary in the existing table "cus_tbl". cus_address is added after cus_surname column and cus_salary is added after cus_age column.

Use the following query to do this:

```
ALTER TABLE cus_tbl
ADD cus_address varchar(100) NOT NULL
AFTER cus_surname,
ADD cus_salary int(100) NOT NULL
AFTER cus_age ;
```

MODIFY column in the table
The MODIFY command is used to change the column definition of the table.

Syntax:

```
ALTER TABLE table_name
```

```
MODIFY column_name column_definition
[ FIRST | AFTER column_name ];
```

Example:

In this example, we modify the column cus_surname to be a data type of varchar(50) and force the column to allow NULL values.

Use the following query to do this:

```
ALTER TABLE cus_tbl
MODIFY cus_surname varchar(50) NULL;
```

DROP column in table

Syntax:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Let's take an example to drop the column name "cus_address" from the table "cus_tbl".

Use the following query to do this:

```
ALTER TABLE cus_tbl
DROP COLUMN cus_address;
```

RENAME column in table

Syntax:

```
ALTER TABLE table_name
CHANGE COLUMN old_name new_name
column_definition
[ FIRST | AFTER column_name ]
```

Example:

In this example, we will change the column name "cus_surname" to "cus_title".

Use the following query to do this:

```
 ALTER TABLE  cus_tbl
CHANGE COLUMN cus_surname cus_title
varchar(20) NOT NULL;
```

RENAME table

Syntax:

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

Example:

In this example, the table name cus_tbl is renamed as cus_table.

```
ALTER TABLE cus_tbl
RENAME TO cus_table;
```

MySQL DROP Table
MYSQL DROP table statement removes the complete data with structure.

Syntax:

```
DROP TABLE  table_name;
```
Example:

This example specifies how to drop a table. In this example, we are dropping the table "cus_tbl".

```
DROP TABLE  cus_tbl;
```

MySQL TRUNCATE Table vs DROP Table
You can also use DROP TABLE command to delete complete table but it will remove complete table data and structure both. You need to re-create the table again if you have to store some data. But in the case of TRUNCATE TABLE, it removes only table data not structure. You don't need to re-create the table again because the table structure already exists.

MySQL INSERT Statement
MySQL INSERT statement is used to insert data in MySQL table within the database. We can insert single or multiple records using a single query in MySQL.

Syntax:

The SQL INSERT INTO command is used to insert data in MySQL table. Following is a generic syntax:

```
INSERT INTO table_name ( field1, field2,...fieldN )
VALUES
( value1, value2,...valueN );
```
Field name is optional. If you want to specify partial values, field name is mandatory.

MySQL UPDATE Query
MySQL UPDATE statement is used to update data of the MySQL table within the database. It is used when you need to modify the table.

Syntax:

Following is a generic syntax of UPDATE command to modify data into the MySQL table:

```
UPDATE table_name SET field1=new-value1, field2=new-value2
[WHERE Clause]
```

MySQL DELETE Statement

MySQL DELETE statement is used to delete data from the MySQL table within the database. By using delete statement, we can delete records on the basis of conditions.

Syntax:

```
DELETE FROM table_name
WHERE
(Condition specified);
```

MySQL SELECT Statement

The MySQL SELECT statement is used to fetch data from the one or more tables in MySQL. We can retrieve records of all fields or specified fields.

Syntax for specified fields:

```
SELECT expressions
FROM tables
[WHERE conditions];
```

Syntax for all fields:

```
SELECT * FROM tables [WHERE conditions];
```

MySQL SELECT Example 3: from multiple tables

MySQL SELECT statement can also be used to retrieve records from multiple tables by using JOIN statement.

Let's take two tables "students" and "officers", having the following data.

MySQL WHERE Clause

MySQL WHERE Clause is used with SELECT, INSERT, UPDATE and DELETE clause to filter the results. It specifies a specific position where you have to do the operation.

MySQL WHERE Clause with AND condition

In this example, we are retrieving data from the table "officers" with AND condition.

Execute the following query:

```
SELECT *
FROM officers
WHERE address = 'Lucknow'
AND officer_id < 5;
```

MySQL Distinct Clause

MySQL DISTINCT clause is used to remove duplicate records from the table and fetch only the unique records. The DISTINCT clause is only used with the SELECT statement.

Syntax:

```
SELECT DISTINCT expressions
FROM tables
[WHERE conditions];
```

MySQL ORDER BY Clause

The MYSQL ORDER BY Clause is used to sort the records in ascending or descending order.

Syntax:

```
SELECT expressions
FROM tables
[WHERE conditions]
ORDER BY expression [ ASC | DESC ];
```

Parameters

expressions: It specifies the columns that you want to retrieve.

tables: It specifies the tables, from where you want to retrieve records. There must be at least one table listed in the FROM clause.

WHERE conditions: It is optional. It specifies conditions that must be fulfilled for the records to be selected.

ASC: It is optional. It sorts the result set in ascending order by expression (default, if no modifier is provider).

DESC: It is also optional. It sorts the result set in descending order by expression.

Note: You can use MySQL ORDER BY clause in a SELECT statement, SELECT LIMIT statement, and DELETE LIMIT statement.

MySQL GROUP BY Clause

The MYSQL GROUP BY Clause is used to collect data from multiple records and group the result by one or more column. It is generally used in a SELECT statement.

You can also use some aggregate functions like COUNT, SUM, MIN, MAX, AVG etc. on the grouped column.

MySQL HAVING Clause

MySQL HAVING Clause is used with GROUP BY clause. It always returns the rows where condition is TRUE.

MySQL LIKE condition
In MySQL, LIKE condition is used to perform pattern matching to find the correct result. It is used in SELECT, INSERT, UPDATE and DELETE statement with the combination of WHERE clause.

Syntax:

expression LIKE pattern [ESCAPE 'escape_character']

MySQL IN Condition
The MySQL IN condition is used to reduce the use of multiple OR conditions in a SELECT, INSERT, UPDATE and DELETE statement.

MySQL IS NULL Condition
MySQL IS NULL condition is used to check if there is a NULL value in the expression. It is used with SELECT, INSERT, UPDATE and DELETE statement.

MySQL BETWEEN Condition
The MYSQL BETWEEN condition specifies how to retrieve values from an expression within a specific range. It is used with SELECT, INSERT, UPDATE and DELETE statement.

JOIN

MySQL JOINS
MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

JION
```
SELECT OrderDetails.OrderID, Orders.OrderDate, Orders.EmployeeID
FROM OrderDetails
```

There are three types of MySQL joins:

MySQL INNER JOIN (or sometimes called simple join)
MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

MySQL Left Outer Join
The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

```
SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
```

```
ON table1.column = table2.column;
```

MySQL Inner JOIN (Simple Join)
The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

Syntax:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

MySQL Right Outer Join
The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```

SQL FULL OUTER JOIN Keyword
The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

The FULL OUTER JOIN keyword returns all matching records from both tables whether the other table matches or not. So, if there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

SQL Self JOIN
A self JOIN is a regular join, but the table is joined with itself.

The SQL UNION Operator
The UNION operator is used to combine the result-set of two or more SELECT statements.

Each SELECT statement within UNION must have the same number of columns
The columns must also have similar data types
The columns in each SELECT statement must also be in the same order

MySQL first function
The MySQL first function is used to return the first value of the selected column. Here, we use limit clause to select first record or more.

```
SELECT ProductName FROM Products LIMIT 100;
```

MySQL Comments

In MySQL, comments can also be placed in the SQL queries. Comments can be of single line or multiple lines.

There are three types in which comments can be written:-

1. Using # symbol

```
# comment goes here
```

2. Using -- symbol

3. Using /* and */ symbol

```
/* comment goes here */
```

Cursors

Oracle uses work areas to execute SQL statements and store processing information. A PL/SQL construct called a cursor lets you name a work area and access its stored information. There are two kinds of cursors: implicit and explicit. PL/SQL implicitly declares a cursor for all SQL data manipulation statements, including queries that return only one row. For queries that return more than one row, you can explicitly declare a cursor to process the rows individually

Multi-row query processing is somewhat like file processing. For example, a COBOL program opens a file, processes records, then closes the file. Likewise, a PL/SQL program opens a cursor, processes rows returned by a query, then closes the cursor. Just as a file pointer marks the current position in an open file, a cursor marks the current position in a result set.

You use the OPEN, FETCH, and CLOSE statements to control a cursor. The OPEN statement executes the query associated with the cursor, identifies the result set, and positions the cursor before the first row. The FETCH statement retrieves the current row and advances the cursor to the next row. When the last row has been processed, the CLOSE statement disables the cursor.

Cursor FOR Loops

In most situations that require an explicit cursor, you can simplify coding by using a cursor FOR loop instead of the OPEN, FETCH, and CLOSE statements. A cursor FOR loop implicitly declares its loop index as a record that represents a row fetched from the database. Next, it opens a cursor, repeatedly fetches rows of values from the result set into fields in the record, then closes the cursor when all rows have been processed. In the following example, the cursor FOR loop implicitly declares emp_rec as a record:

```
DECLARE
   CURSOR c1 IS
      SELECT ename, sal, hiredate, deptno FROM emp;
   ...
BEGIN
```

```
   FOR emp_rec IN c1 LOOP
      ...
      salary_total :=  salary_total + emp_rec.sal;
   END LOOP;
```

To reference individual fields in the record, you use dot notation, in which a dot (.) serves as the component selector.

MySQL Functions
Creating a function
In MySQL, Function can also be created. A function always returns a value using the return statement. The function can be used in SQL queries.

Literals (Constants)
In MySQL, literals are the constant. Following are the types of literal:-

1. String Literal
In MySQL, string literals are in single quotes (') or double quotes (").

2. Number Literals
In MySQL, number literals are positive or negative numbers.

Literals: Date and Time
In MySQL programming, Date and Time literals are in the form of strings or numbers.

Creating a procedure
In MySQL, a procedure can also be created. A procedure can return one or more than one value through parameters or may not return at all. The procedure can be used in SQL queries.

What is a Stored Procedure?
A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

In MySQL, a procedure can also be dropped. When a procedure id dropped, it is removed from the database.

```
DELIMITER $$
CREATE PROCEDURE get_student()
BEGIN
SELECT * FROM table1;
```

```
END$$
```

Syntax:
Drop procedure[IF EXISTS] procedure_name;
Parameter:
procedure_name: name of the procedure to be dropped.

MySQL Trigger

Trigger: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.
In MySQL, trigger can also be created. There are 6 type of triggers that can be made they are:-

After/Before insert
After/Before update
After/Before delete

Syntax

```
CREATE TRIGGER trigger_name
    AFTER/BEFORE INSERT
         ON table_name FOR EACH ROW
         BEGIN
        --variable declarations
        --trigger code
        END;
```

DATABASE (DBMS)

Database: Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc. For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

Database Management System: The software which is used to manage database is called Database Management System (DBMS). For Example, MySQL, Oracle etc. are popular commercial DBMS used in different applications

DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.

DBMS Architecture
The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
The client/server architecture consists of many PCs and a workstation which are

connected via the network.
DBMS architecture depends upon how users are connected to the database to get their request done.

Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: 2-tier architecture and 3-tier architecture.

1-Tier Architecture
In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.
2-Tier Architecture
The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: ODBC, JDBC are used.

3-Tier Architecture
The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.

Database Language
A DBMS has appropriate languages and interfaces to express database queries and updates.
Database languages can be used to read, store and update the data in the database.

1. Data Definition Language
DDL stands for Data Definition Language. It is used to define database structure or pattern.
It is used to create schema, tables, indexes, constraints, etc. in the database.
Using the DDL statements, you can create the skeleton of the database.
Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.
Here are some tasks that come under DDL:

Create: It is used to create objects in the database.
Alter: It is used to alter the structure of the database.
Drop: It is used to delete objects from the database.
Truncate: It is used to remove all records from a table.
Rename: It is used to rename an object.
Comment: It is used to comment on the data dictionary.
These commands are used to update the database schema that's why they come under Data definition language.

2. Data Manipulation Language

DML stands for Data Manipulation Language. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

Select: It is used to retrieve data from a database.
Insert: It is used to insert data into a table.
Update: It is used to update existing data within a table.
Delete: It is used to delete all records from a table.
Merge: It performs UPSERT operation, i.e., insert or update operations.
Call: It is used to call a structured query language or a Java subprogram.
Explain Plan: It has the parameter of explaining data.
Lock Table: It controls concurrency.
3. Data Control Language
DCL stands for Data Control Language. It is used to retrieve the stored or saved data.
The DCL execution is transactional. It also has rollback parameters.
(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

Grant: It is used to give user access privileges to a database.
Revoke: It is used to take back permissions from the user.
There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

4. Transaction Control Language
TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

Commit: It is used to save the transaction on the database.
Rollback: It is used to restore the database to original since the last Commit.

ER model
ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

Entity:
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

Weak Entity

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

Attribute
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.

Composite Attribute

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

Relationship
A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

Types of relationship are as follows:

a. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

DBMS ER model concept
b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of

an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.

d. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

Keys
Keys play an important role in the relational database.
It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.
For example: In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

DBMS Keys
Types of key:

DBMS Keys
1. Primary key
It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.
In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique.
For each entity, selection of the primary key is based on requirement and developers.

DBMS Keys

2. Candidate key

A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.

The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.

DBMS Keys

3. Super Key

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same.

Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

Foreign keys are the column of the table which is used to point to the primary key of another table.

In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.

We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table.

Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

Specialization

Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.

Aggregation

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

Entity integrity constraints

The entity integrity constraint states that primary key value can't be null.

SQL
SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).
It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.
SQL allows users to query the database in a number of ways, using English-like statements.

Normalization
Normalization is the process of organizing the data in the database.
Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
Normalization divides the larger table into the smaller table and links them using relationship.
The normal form is used to reduce redundancy from the database table.

First Normal Form (1NF)
A relation will be 1NF if it contains an atomic value.
It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
First normal form disallows the multi-valued attribute, composite attribute, and their combinations.
Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

Second Normal Form (2NF)
In the 2NF, relational must be in 1NF.
In the second normal form, all non-key attributes are fully functional dependent on the primary key

Third Normal Form (3NF)
A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

Boyce Codd normal form (BCNF)
BCNF is the advance version of 3NF. It is stricter than 3NF.
A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Fourth normal form (4NF)
A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.
Example

Fifth normal form (5NF)
A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
5NF is also known as Project-join normal form (PJ/NF).

Relational Decomposition
When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
In a database, it breaks the table into multiple tables.
If the relation has no proper decomposition, then it may lead to problems like loss of information.
Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

SQL Injection
SQL injection is a code injection technique that might destroy your database.

SQL injection is one of the most common web hacking techniques.

SQL injection is the placement of malicious code in SQL statements, via web page input.

SQL CREATE INDEX Statement
The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

REATE INDEX Syntax
Creates an index on a table. Duplicate values are allowed:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

CREATE UNIQUE INDEX Syntax
Creates a unique index on a table. Duplicate values are not allowed:

```
CREATE UNIQUE INDEX index_name
```

```
ON table_name (column1, column2, ...);
```

Drop INDEX
SQL Server:

```
DROP INDEX table_name.index_name;
```

So, why do you need to index your tables?
Because without an index the SQL server has to scan the entire table to return the requested data. It is like the index page in a book. You check within the index for the keyword you want to learn about. From that point forward, you jump directly to the page where the content belongs, instead of scanning page by page for the material you want to read.

SQL Index Types
There are two main index types: Clustered index and Non-Clustered index.

A clustered index alters the way that the rows are physically stored. When you create a clustered index on a column (or a number of columns), the SQL server sorts the table’s rows by that column(s).

It is like a dictionary, where all words are sorted in an alphabetical order. Note, that only one clustered index can be created per table. It alters the way the table is physically stored, it couldn’t be otherwise.

In the example below, all rows are sorted by computer_id, as a clustered index on the computer_id column has been created.

A non-clustered index, on the other hand, does not alter the way the rows are stored in the table. Instead, it creates a completely different object within the table, that contains the column(s) selected for indexing and a pointer back to the table’s rows containing the data.

It is like an index in the last pages of a book. All keywords are sorted and contain a reference back to the appropriate page number. A non-clustered index on the computer_id column, in the previous example, would look like the table below:

Note, that SQL server sorts the indexes efficiently by using a B-tree structure. This is a tree data structure that allows SQL Server to keep data sorted, to allow searches, sequential access, insertions and deletions, in a logarithmic amortized time. This methodology minimizes the number of pages accessed, in order to locate the desired index key, therefore resulting in an improved performance.

Relation between clustered and non-clustered indexes
As explained above, a non-clustered index contains a pointer back to the rowID (RID), of the table, in order to relate the index’s column with the rest of the columns in a row.

But this is not always the case:

If a clustered index already exists in the table, the non-clustered index uses the clustered index’s key as the row locator, instead of the RID reference.

Codd's Rules for RDBMS

Rule Zero

The system must qualify as relational, as a database, and as a management system. For a system to qualify as a relational database management system (RDBMS), that system must use its relational facilities (exclusively) to manage the database.
The other 12 rules derive from this rule. The rules are as follows :
Rule 1 : The information rule: All information in the database is to be represented in one and only one way, namely by values in column positions within rows of tables.

Rule 2 : The guaranteed access rule: All data must be accessible. This rule is essentially a restatement of the fundamental requirement for primary keys. It says that every individual scalar value in the database must be logically addressable by specifying the name of the containing table, the name of the containing column and the primary key value of the containing row.

Rule 3 : Systematic treatment of null values: The DBMS must allow each field to remain null (or empty). Specifically, it must support a representation of "missing information and inapplicable information" that is systematic, distinct from all regular values (for example, "distinct from zero or any other number", in the case of numeric values), and independent of data type. It is also implied that such representations must be manipulated by the DBMS in a systematic way.

Rule 4 : Active online catalog based on the relational model: The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language. That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.

Rule 5 : The comprehensive data sub language rule: The system must support at least one relational language that

1. Has a linear syntax

2. Can be used both interactively and within application programs,

3. Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).

Rule 6 : The view updating rule: All views those can be updated theoretically,

must be updated by the system.

Rule 7 : High-level insert, update, and delete: The system must support set-at-a-time insert, update, and delete operators. This means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8 : Physical data independence: Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.

Rule 9 : Logical data independence: Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure. Logical data independence is more difficult to achieve than physical data independence.

Rule 10 : Integrity independence: Integrity constraints must be specified separately from application programs and stored in the catalog. It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.

Rule 11 : Distribution independence: The distribution of portions of the database to various locations should be invisible to users of the database. Existing applications should continue to operate successfully :

1. when a distributed version of the DBMS is first introduced; and

2. when existing distributed data are redistributed around the system.

Rule 12: The non-subversion rule: If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint.

MongoDB is a No SQL database. It is an open-source, cross-platform, document-oriented database written in C++.

Purpose of building MongoDB
It may be a very genuine question that - "what was the need of MongoDB although there were many databases in action?"

There is a simple answer:

All the modern applications require big data, fast features development, flexible deployment, and the older database systems not competent enough, so the MongoDB was needed.

NoSQL Database
NoSQL Database is used to refer a non-SQL or non relational database.

It provides a mechanism for storage and retrieval of data other than tabular relations model used in relational databases. NoSQL database doesn't use tables for storing data. It is generally used to store big data and real-time web applications.

MongoDB Advantages
MongoDB is schema less. It is a document database in which one collection holds different documents.
There may be difference between number of fields, content and size of the document from one to other.
Structure of a single object is clear in MongoDB.
There are no complex joins in MongoDB.
MongoDB provides the facility of deep query because it supports a powerful dynamic query on documents.
It is very easy to scale.
It uses internal memory for storing working sets and this is the reason of its fast access.