In This Chapter

CHAPTER 13

MySQL Revision Tour

13.1 Introduction

A database system is basically a computer based record keeping system. The collection of data, usually referred to as the *database*, contains information about one particular enterprise. In a typical file-processing system, permanent records are stored in various files. A number of different application programs are written to extract records from files and add records to the appropriate files. But this scheme has a number of major limitations and disadvantages, such as data redundancy (duplication of data), data inconsistency, unsharable data, unstandardized data, insecure data, incorrect data etc. A *database management system* is answer to all these problems as it provides a centralized control of the data.

Various advantages of database systems are :

- Database systems reduce data redundancy (data duplication) to a large extent.
- Database systems control data inconsistency to a large extent.
- Databases facilitate sharing of data.
- Databases enforce standards.
- Centralized databases can ensure data security.
- Integrity can be maintained through databases.

13.2 Different Data Models

A data model refers to a set of concepts to describe the structure of a database, and certain *constraints* (restrictions) that the database should obey.

The *four* data models that are used for database management are :

- Relational data model,
- Network data model,
- Hierarchical data model,
- Object Oriented data model.

(*i*) *The Relational Data Model.* In relational data model, the data is organized into tables (*i.e.*, rows and columns). These tables are called *relations*. A row in a table represents a *relationship* among a set of values. Since a table is a collection of such relationships, it is generally referred to using the mathematical term *relation*, from which the relational data model derives its name.

(*ii*) *The Network Data Model.* The network model differs from the relational model in that data is represented by collections of *records* and relationships among data are represented by *links*. In a network database, the collections of records are connected to one another by means of *links*. *A record is a collection of fields* (*attributes*), *each of which contains only one data value*. By a *link* we mean that it is an association between precisely two records.

(*iii*) *The Hierarchical Data Model.* In the network model, the data is represented by collections of records and relationships among data and are represented by *links*. This is true of hierarchical model as well. The only difference is that in the hierarchical model, records are organized as *trees* rather than arbitrary graphs.

In hierarchical model, the records have Parent-Child Relationship in 1:N form. The record on the 1 side is called the *parent* and the record on the N side is called the *child*.

(*iv*) *Object Oriented Data Model.* In object oriented data model, data and associated operations are represented by *objects*. An object is an identifiable entity with some characteristics and behaviour. Similar objects are conceptually collected together into meaningful groups called *classes e.g.*, *'Mohan is a student'* means *Mohan object* belongs to *student class*. Objects of the same classes have common attributes, behaviours and relationships with other objects. Like the network model, the relationships among objects are specified via a "physical" link (pointer) between objects.

The essential features of OO data model are :

- ***Object identity*** — The ability of the system to distinguish between two different objects that have the same state.
- ***Encapsulation*** — The ability to wrap up data and associated operations under one unit. Encapsulation requires that all access or interaction with objects be done by invoking the services provided by their external interface.
- ***Complex state*** — The ability to define data types using primitive types using which complex objects can be built.
- ***Type extensibility*** — The ability to define new data types from previously defined types by enhancing or changing the structure or behaviour of the types. Type inheritance is a mechanism used to define new types by enhancing already existing behaviour.
- ***Genericity*** — It means that the types of the data model with which the object query language collaborates must be *generic i.e.*, as a new type is added to the system, it must be compatible and queriable.

13.3 Relational Model Terminology

Let us now revise different terms used in relational model.

- *Relation* — A table storing logically related data ; data must be atomic in a cell ; all rows of this table are distinct ; ordering of rows and columns is immaterial.
- *Domain* — This is a pool of values from which the actual values appearing in a given column are drawn.
- *Tuple* — A row of a relation is generally referred to as a *tuple*.
- *Attribute* — A column of a relation is generally referred to as an *attribute*.
- *Degree* — This refers to the number of attributes in a relation.
- *Cardinality* — This refers to the number of tuples in a relation.
- *View* — It is a virtual table that does not really exist in its own right but is instead derived from one or more underlying base table(s).
- *Primary Key* — This refers to a set of one or more attributes that can uniquely identify tuples within the relation.
- *Candidate Key* — All attribute combinations inside a relation that can serve as primary key are *candidate keys* as these are candidates for primary key position.
- *Alternate Key* — A candidate key that is not primary key, is called an *alternate key*.
- *Foreign Key* — A non-key attribute, whose values are derived from the primary key of some other table, is known as *foreign key* in its current table.

Referential Integrity

A referential integrity is a system of rules that a DBMS uses to ensure that relationships between records in related tables are valid, and that users don't accidentally delete or change related data.

You can set referential integrity when all of the following conditions are met :

- The matching field from the primary table is a primary key or has a unique index.
- The related fields have the same data type.
- Both tables belong to the same database. Referential integrity can't be enforced for linked table from databases in other formats.

When referential integrity is enforced, you must observe the following rules :

- You can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table. However, you can enter a Null value in the foreign key, specifying that the records are unrelated.
- You can't delete a record from a primary table if matching records exist in a related table.
- You can't change a primary key value in the primary table, if that record has related records.

13.4 Introduction To MySQL

MySQL is a freely available open source *Relational Database Management System* (RDBMS) that uses Structured Query Language (SQL). It is downloadable from site www.mysql.org In a MySQL database, information is stored in *Tables*. A single MySQL database can contain many tables at once and store thousands of individual records. MySQL provides you with a rich set of features that support a secure environment for storing, maintaining, and accessing data. MySQL is a fast, reliable, scalable alternative to many of the commercial RDBMSs available today.

Brief History of MySQL

MySQL was created and is supported by *MySQL AB*, a company based in Sweden (www.mysql.com). This company is now a subsidiary of *Sun Microsystems*, which holds the copyright to most of the codebase. On April 20th, 2009 *Oracle Corp.*, which develops and sells the proprietary *Oracle database*, announced a deal to acquire *Sun Microsystems*.

The chief inventor of MySQL was *Michael Widenius* (a.k.a. *Monty*). MySQL has been named after Monty's daughter *My*. The logo of MySQL, the dolphin, is named as *"Sakila"*.

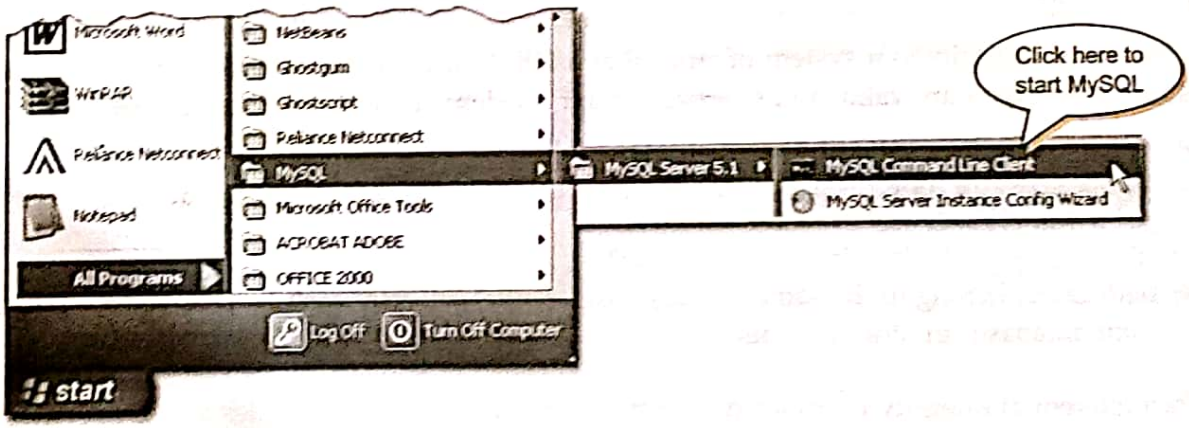
MySQL Database System

MySQL database system refers to the combination of a MySQL server instance and a MySQL database. MySQL operates using client/server architecture in which the server runs on the machine containing the databases and clients connect to the server over a network.

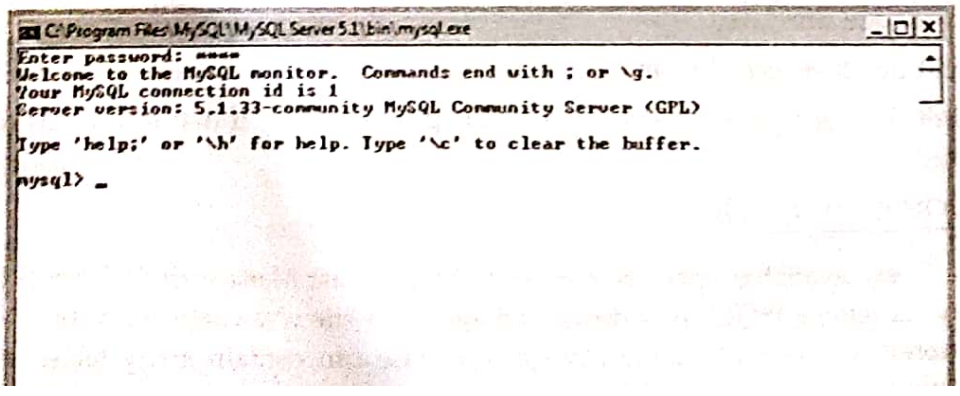
- **The server** (MySQL server) listens for client requests coming in over the network and accesses database contents according to those requests and provides that to the clients.
- **Clients** are programs that connect to the database server and issue queries in a pre-specified format. MySQL is compatible with the standards based SQL (Structured Query Language).

13.5 Starting MySQL

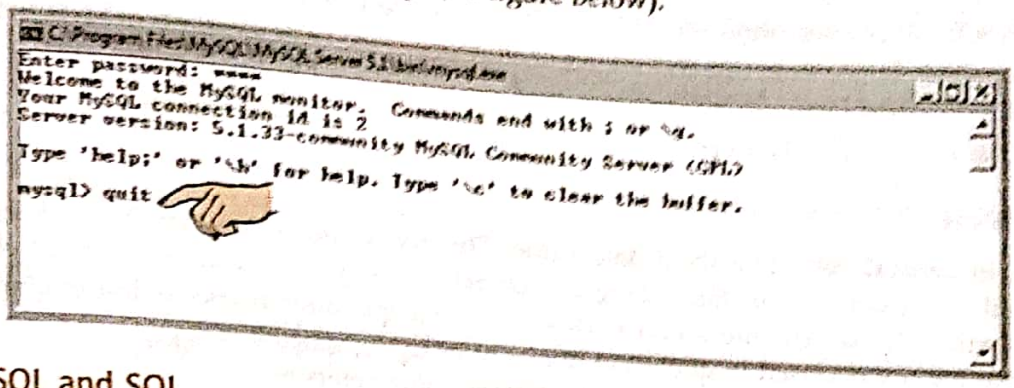
Starting MySQL is similar to the way you start other applications in Windows platform. Make sure that MySQL Server is installed on your machines. Once it is installed, you need to click at **Start → All Programs → MySQL → MySQL Server → MySQL Command Line Client** (see figure below)



It will start the MySQL client where you have to specify the password. Ask your teacher about the password for the MySQL installation in your lab.



Now you can start working in MySQL. When you are through with your work, you can quit from MySQL by typing Quit at the mysql> prompt (see figure below).



13.6 MySQL and SQL

In order to access data within the MySQL database, all programs and users must use, *Structured Query Language* (SQL). SQL is the set of commands that is recognised by nearly all RDBMSs.

The Structured Query Language (SQL) is a language that enables you to create and operate on relational databases, which are sets of related information stored in tables.

13.6.1 Classification of SQL Statements

SQL provides many different types of commands used for different purposes. SQL commands can be mainly divided into following categories :

(*i*) **Data Definition Language (DDL) Commands.** Commands that allow you to perform tasks related to data definition *e.g.*,

- creating, altering and dropping.
- maintenance commands
- granting and revoking privileges and roles.

(*ii*) **Data Manipulation Language (DML) Commands.** Commands that allow you to perform data manipulation *e.g.*, retrieval, insertion, deletion and modification of data stored in a database.

(*iii*) **Transaction Control Language (TCL) Commands.** Commands that allow you to manage and control the transactions (a transaction is one complete unit of work involving many steps), *e.g.*,

- making changes to database, permanent
- undoing changes to database, permanent
- creating savepoints
- setting properties for current transactions

There are other categories of SQL commands also but above three categories of commands are mainly used by learners.

Let Us Revise

- *MySQL is a free, open-source Relational Database Management System.*
- *A MySQL database system consists of a MySQL server instance and a MySQL database.*
- *The DDL commands are used to define or redefine schema objects.*
- *The DML commands are used to manipulate data in existing schema objects.*
- *The TCL commands are used to manage transactions.*
- *A transaction is one complete unit of work.*

13.7 Some MySQL SQL Elements

The MySQL implementation of SQL has certain elements that play an important role in defining/querying a database. In this section, we shall revise some basic elements of MySQL SQL that you must be aware of. These basic elements are :

(*i*) Literals (*ii*) Datatypes (*iii*) Nulls (*iv*) Comments

13.7.1 Literals

Literals, in general, refer to a fixed data value. This fixed data value may be of ***character*** type or ***numeric*** literal. For example, *'Synthia'*, *'Ekagra'*, *'Raunak Raj Singh'*, *'8'* and *'305'* are all ***character text literals***. Notice that all character literals are enclosed in single quotation marks or double quotation marks. Characters that are not enclosed in quotation marks refer to the schema object names *e.g., value* refers to a schema object whereas *'value'* refers to a character literal.

NOTE

A text literal can have maximum length of 4000 bytes in MySQL 5.1.

To store an apostrophe in a text literal, you should use \' (backslash followed by apostrophe *e.g.,* to store *Kush's* in a text literal, you should write it as 'Kush\'s').

Numbers that are not enclosed in quotation marks are numeric literals *e.g.,* 22, 18, 1997, 2003 are all ***numeric literals***.

NOTE

A numeric literal can store a maximum of 53 digits of precision.

Numeric literals can either be **integer literals** *i.e.,* without any decimal or be **real literals** *i.e.,* with a decimal point *e.g., 17* is an integer literal but *17.0* and *17.5* are *real literals*.

13.7.2 Data Types

Data types are means to identify the type of data and associated operations for handling it. A value's *datatype* associates a fixed set of properties with the value. These properties cause different treatment of datatypes. For example, you can add values of NUMBER datatypes, but you can not add values of CHAR or VARCHAR types. Internal datatypes supported by MySQL include the ones given here.

MySQL uses many different data types, divided into *three* categories :

- Numeric
- Date and time, and
- String types.

Numeric Data Types

MySQL uses all the standard ANSI SQL numeric data types.

The following list shows the common numeric data types and their descriptions.

INT	A *normal-sized integer that can be signed or unsigned*. If signed, the allowable range is from –2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
TINYINT	*A very small integer that can be signed or unsigned*. If signed, the allowable range is from –128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
SMALLINT	*A small integer that can be signed or unsigned*. If signed, the allowable range is from –32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
MEDIUMINT	*A medium-sized integer that can be signed or unsigned*. If signed, the allowable range is from –8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
BIGINT	*A large integer that can be signed or unsigned*. If signed, the allowable range is from –9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 11 digits.

FLOAT(M,D)	*A floating-point number that cannot be unsigned.* You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
DOUBLE(M,D)	*A double precision floating-point number that cannot be unsigned.* You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
DECIMAL(M,D)	*An unpacked floating-point number that cannot be unsigned.* In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types

The MySQL date and time datatypes are :

DATE	*A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31.* For example, December 30th, 1973 would be stored as 1973-12-30.
DATETIME	*A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59.* For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
TIMESTAMP	*A timestamp between midnight, January 1, 1970 and sometime in 2037.* This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
TIME	*Stores the time in HH:MM:SS format.*
YEAR(M)	*Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.*

String/Text Types

Most data that you use in a database is in string format. This list describes the common string datatypes in MySQL.

CHAR(M)	*A fixed-length string between 1 and 255 characters in length* (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
VARCHAR(M)	*A variable-length string between 1 and 255 characters in length;* for example VARCHAR(25). You must define a length when creating a VARCHAR field.
BLOB or TEXT	*A field with a maximum length of 65535 characters.* BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.
TINYBLOB or TINYTEXT	*A BLOB or TEXT column with a maximum length of 255 characters.* You do not specify a length with TINYBLOB or TINYTEXT.
MEDIUMBLOB or MEDIUMTEXT	*A BLOB or TEXT column with a maximum length of 16777215 characters.* You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
LONGBLOB or LONGTEXT	*A BLOB or TEXT column with a maximum length of 4294967295 characters.* You do not specify a length with LONGBLOB or LONGTEXT.
ENUM	*An enumeration is a fancy term for list.* When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

Difference between Char and Varchar

The difference between CHAR and VARCHAR is that of *fixed length* and *variable length*. The CHAR datatype specifies a *fixed length* character string. When a column is given datatype as CHAR(*n*), then MySQL ensures that all values stored in that column have this length *i.e.*, *n* bytes. If a value is shorter than this length *n* then blanks are added, but the size of value remains *n* bytes.

VARCHAR, on the other hand, specifies a variable length string. When a column is given datatype as VARCHAR(*n*), then the maximum size a value in this column can have is *n* bytes. Each value that is stored in this column stores exactly as you specify it *i.e.*, no blanks are added if the length is shorter than maximum length *n*. However, if you exceed the maximum length *n*, then an error message is displayed.

13.7.3 Null Values

If a column in a row has no value, then column is said to be *null*, or to contain a null. Nulls can appear in columns of any data type provided they are not restricted by NOT NULL or PRIMARY KEY integrity constraints[1]. You should use a *null* value when the actual value is not known or when a value would not be meaningful.

One thing that you should make sure is, not to use null to represent a value of zero, because they are not equivalent. Any arithmetic expression containing a null always evaluates to null. For example, null added to 10 is null. In fact, all operators (except concatenation) return null when given a null operand.

NOTE

Any arithmetic expression containing a null, always evaluates to null.

13.7.4 Comments

A comment is a text that is not executed ; it is only for documentation purpose.

Comments can be written in MySQL in following three ways :

(*i*) Comment enclosed in /*.....*/. This is multiline comment.

(*ii*) Comment beginning with two hyphens followed by a space (--). This is a single line comment.

(*iii*) Comment beginning with a # symbol. This is a single line comment.

13.8 Accessing Database in MySQL

Before you start writing SQL commands or making queries upon the data in tables of a database, you need to open the database for use. For this, after logging into MySQL, you need to issue a command

```
Use <databasename> ;
```

For example, we want to work on our sample database namely *menagerie*, so we shall write the following command after logging in *MySQL*.

```
mysql> USE menagerie ;
Database changed
mysql>
```

13.9 Creating Tables in MySQL

Tables are defined with the CREATE TABLE command. When a table is created, its columns are named, data types and sizes are supplied for each column. Each table must have at least one column.

1. A constraint is a condition applicable on column(s) of a table.

The syntax of CREATE TABLE command is :

Syntax

```
CREATE TABLE <table-name>
    (<column name> <data type> [ (<size>)],
    <columnname> <data type> [ ( <size> ) ... ]) ;
```

To create an *employee* table whose schema is as follows :

```
employee (ecode, ename, sex, grade, gross)
```

the *SQL* command will be

```
CREATE TABLE employee
(   ecode    integer,
    ename    char(20),
    sex      char(1),
    grade    char(2),
    gross    decimal  ) ;
```

NOTE

Before issuing a CREATE TABLE command, make sure that its parent database has been opened using USE <database name> command.

13.10 Inserting Data Into Table

The rows (tuples) are added to relations using INSERT command of *SQL*. In its simplest form, INSERT takes the following syntax :

Syntax

```
INSERT INTO <tablename> [ <column list> ]
VALUES ( <value>, <value> ... ) ;
```

For example, to enter a row into *employee* table (defined earlier), you could use the following statement :

```
INSERT INTO employee
VALUES (1001, 'Ravi', 'M', 'E4', 4670.00) ;
```

See the order of values matches the order of columns in the CREATE TABLE command of *employee*. The same can be done with an alternate command as shown below :

```
INSERT INTO employee (ecode, ename, sex, grade, gross)
VALUES (1001, 'Ravi', 'M', 'E4', 4670.00) ;
```

The INSERT statement adds a new row to *employee* giving a value for every column in the row. Note that the data values are in the same order as the column names in the table. Data can be added only to some columns in a row by specifying the columns and their data.

For instance, if you want to insert only *ecode*, *ename* and *sex* columns, you use the command :

```
INSERT INTO employee (ecode, ename, sex)
VALUES (2014, 'Manju', 'F') ;
```

The columns that are not listed in the INSERT command will have their default value, if it is defined for them, otherwise, NULL value.

If any other column (that does not have a **default value** and is defined NOT NULL) is skipped or omitted, an error message is generated and the row is not added.

NOTE

In an INSERT statement, only those columns can be omitted that have either default value defined or they allow NULL values.

Inserting NULL values

To insert value NULL in a specific column, you can type NULL without quotes and NULL will be inserted in that column. Consider the following statement :

```
INSERT INTO EMPL (Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno)
VALUES (8100, 'YASH', 'ANALYST', NULL, '10-MAY-03', 6000, NULL, 20) ;
```

See, for **Mgr** and **Comm** columns, NULL values have been inserted.

Inserting Dates

Dates are by default entered in 'YYYY-MM-DD' format *i.e.*, first four digits depicting *year*, followed by a hyphen, followed by 2 digits of *month*, followed by a hyphen and a two digit day. All this is enclosed in single quotes.

13.11 Making Simple Queries Through Select Command

The SELECT statement is used to pull information from a table. The general form of the statement is :

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy ;
```

13.11.1 Selecting All Data

The simplest form of SELECT retrieves everything from a table. You just need to specify asterisk in the *select-list*(what_to_select), *e.g.*, :

```
mysql> SELECT * FROM pet;
```

13.11.2 Selecting Particular Rows

You can select particular rows from a table by specify filtering condition through WHERE clause of the SELECT statement, *e.g.*,

1. Select all pets with gender(sex) as *male*("m").

```
mysql> SELECT * FROM pet
       WHERE sex = 'm';
```

2. Select all pets that were born on or after Jan 1, 2014.

```
mysql> SELECT * FROM pet
       WHERE birth >= '2014-1-1';
```

← *date in yyyy-mm-dd format*

3. Select all female-dogs.

```
mysql> SELECT * FROM pet
       WHERE species = 'dog' AND sex = 'f';
```

4. Select all snakes or birds.

```
mysql> SELECT * FROM pet
       WHERE species = 'snake' OR species = 'bird';
```

5. Select all male cats.

```
mysql> SELECT * FROM pet
       WHERE (species = 'cat' AND sex = 'm') ;
```

13.11.3 Selecting Particular Columns

You can select particular columns by specifying column-names (*i.e.*, attributes) in the select-list of the SELECT command, *e.g.*,

1. Display names and birth-dates of all pets.

```
mysql> SELECT name, birth FROM pet ;
```

2. Display owners of pets born after Dec 2013.

```
mysql> SELECT owner FROM pet
       WHERE birth >'2013-12-31' ;
```

← *date in yyyy-mm-dd format*

13.11.4 Eliminating Redundant Data (with Keyword DISTINCT)

By default, data is selected from all the rows of the table, even if the data appearing in the result gets duplicated. The DISTINCT keyword eliminates duplicate rows from the results of a SELECT statement. *For example,*

1. Display names of all pet-owners(non-redundant).

```
mysql> SELECT DISTINCT owner FROM pet;
```

2. Display distinct species of pets from table pet.

```
mysql> SELECT DISTINCT(species) FROM pet.
```

13.11.5 Selecting From All the Rows – ALL keyword

If in place of keyword DISTINCT, you give keyword ALL then the result retains the duplicate output rows. It is just the same as when you specify neither DISTINCT nor ALL ; ALL is essentially a clarifier rather than a functional argument. Thus if you give

```
SELECT ALL city FROM suppliers ;
```

it will give values of *city* column from every row of the table without considering the duplicate entries.

13.11.6 Viewing Structure of a Table

If you want to know the structure of a table, you can use *Describe* or *Desc* command as per following syntax :

```
DESCRIBE | DESC <table name> ;
```

For instance, the commands : DESCRIBE pet ; or DESC pet ; will display the structure of table *pet*.

```
mysql> DESC pet ;
```

13.11.7 Performing Simple Calculations

To perform simple calculations, you can write the expression/formula to be calculated next to keyword SELECT, *e.g.*,

1. To calculate 3.14159*6*6

```
mysql> SELECT 3.14159*6*6 ;
```

2. To obtain current system date

```
mysql> SELECT curdate();
```

13.11.8 Using Column Aliases

The columns that you select in a query can be given a different name *i.e.*, **column alias name** for output purposes.

As per following syntax :

```
Select <columnname> AS [columnalias] [, <columnname> AS [columnalias]]
    .....
From <tablename> ;
```

Columnalias name of column type

For example,

```
mysql>  SELECT date, type AS "Event Type"
    ->  FROM event ;
```

13.11.9 Condition Based on a Range

The BETWEEN operator defines a range of values that the column values must fall in to make the condition true. The range includes both lower value and the upper value.

For example, to list the items whose *QOH* falls between 30 to 50 (both inclusive), the command would be :

```
SELECT icode, descp, QOH
FROM items
WHERE QOH BETWEEN 30 AND 50 ;
```

13.11.10 Condition Based on a List

To specify a list of values, IN operator is used. The IN operator selects values that match any value in a given list of values. For example, to display a list of members from 'DELHI', 'MUMBAI', 'CHENNAI' or 'BANGALORE' cities, you may give

```
SELECT * FROM members
WHERE city IN ('DELHI', 'MUMBAI', 'CHENNAI', 'BANGALORE') ;
```

The NOT IN operator finds rows that do not match in the list. So if you write

```
SELECT * FROM members
WHERE city NOT IN ('DELHI', 'MUMBAI', 'CHENNAI') ;
```

it will list members not from the cities mentioned in the list.

13.11.11 Condition Based on Pattern Matches

SQL also includes a string-matching operator, LIKE, for comparisons on character strings using patterns. Patterns are described using two special wildcard characters :

- **percent (%).** The % character matches any substring.
- **underscore (_).** The _ character matches any character.

The LIKE keyword is used to select rows containing columns that match a wildcard pattern.

Examples :

1. To list members which are in areas with pin codes starting with 13, the command is :

```
SELECT firstname, lastname, city
FROM members
WHERE pin LIKE '13%' ;
```

2. To list names of pets who have names ending with 'y', the command would be :

```
SELECT name
FROM emp
WHERE name LIKE '%y' ;
```

13.11.12 Searching for NULL

The NULL value in a column can be searched for in a table using IS NULL in the WHERE clause. (Relational operators like =, <> etc. can't be used with NULL). For example, to list details of all employees whose departments contain NULL (*i.e.*, novalue), you use the command :

```
SELECT empno, empname, job
FROM emp WHERE DeptNo IS NULL ;
```

Non-NULL values in a table can be listed using IS NOT NULL.

13.11.13 Sorting Results – ORDER BY clause

Whenever a SELECT query is executed, the resulting rows emerge in a predecided order. You can sort the results or a query in a specific order using ORDER BY clause. The ORDER BY clause allows sorting of query results by one or more columns. The sorting can be done either in *ascending* or *descending* order, the default order is *ascending*. The data in the table is not sorted ; only the results that appear on the screen are sorted.

The ORDER BY clause is used as :

```
SELECT <column name> [, <column name> , ... ]
FROM <table name>
[WHERE <predicate> ][ORDER BY <column name> ] ;
```

for example, to display the list of employees in the alphabetical order of their names, you use the command :

```
SELECT * FROM employee
ORDER BY ename ;
```

To display the list of students having aggregate more than 400 in the alphabetical order of their names, you may give the command :

```
SELECT name, aggregate FROM student
WHERE aggregate > 400
ORDER BY name ;
```

Let Us Revise

- *The basic elements of MySQL SQL are : literals, datatypes, nulls and comments.*
- *Literals are fixed data values.*
- *Data types of MySQL SQL include : NUMERIC (INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT, FLOAT(M, D), DOUBLE(M, D) AND DECIMAL(M, D), DATE and TIME (DATE, DATETIME, TIMESTAMP, TIME and YEAR(M) and STRING (CHAR(M), VARCHAR(M), BLOB or TEXT, TINYBLOB or TINYTEXT, MEDIUMBLOB or MEDIUMTEXT and ENUM).*
- *Column having no value is said to have NULL value.*
- *The SELECT command of SQL lets you make queries on the database.*
- *The DISTINCT keyword eliminates redundant data.*
- *To perform calculations, the expressions can be written along with SELECT without specifying any table name.*
- *A small work table **Dummy** can be used for making simple calculations.*
- *The **curdate** pseudo-column returns the current system data.*
- *The WHERE clause is used to select specific rows.*
- *The logical operators OR (||), AND (&&) and NOT (!) are used to connect search conditions in the WHERE clause.*
- *The BETWEEN operator is used for making range checks in queries.*
- *The IN operator is used for selecting values from a list of values.*
- *The LIKE operator is used for making character comparisons using strings.*
- *The null value in a column can be searched for in a table using IS NULL in the WHERE clause.*
- *The ORDER BY clause is used to sort the results of a query.*

13.12 MySQL Functions

A function is a special type of predefined command set that performs some operation and returns a single value. Functions operate on zero, one, two or more values that are provided to them. The values that are provided to functions are called **parameters** or **arguments.**

The MySQL functions have been categorised into various categories, such as *String functions, Mathematical functions, Date and Time functions* and so on. Although MySQL offers a big bouquet of functions, yet in this discussion we shall remain limited to the functions as mentioned in your syllabus.

13.12.1 String Functions

The string functions of MySQL can manipulate the text string in many ways. Some commonly used string functions are being discussed below.

	Function	Description	Examples
1.	CHAR( )	Returns the character for each integer passed	1. SELECT CHAR (70, 65, 67, 69) ; 2. SELECT CHAR (65, 67.3, '69.3) ;
2.	CONCAT( )	Returns concatenated string	SELECT CONCAT(name, aggregate) AS "Name Marks" FROM student WHERE age = 14 OR age = 16 ;
3.	LOWER()/LCASE( )	Returns the argument in lowercase	SELECT LOWER('MR. OBAMA') AS "LowerName1", LOWER('Ms. Gandhi') AS "LowerName2" ;
4.	SUBSTRING( ), SUBSTR()	Returns the substring as specified	1. SELECT SUBSTR('ABCDEFG', 3, 4) "Subs" ; 2. SELECT SUBSTR ('ABCDEFG', -5, 4) "Subs"
5.	UPPER()/UCASE( )	Converts to uppercase	SELECT UPPER('Large') "Uppercase" ; or SELECT UCASE('Large') "Uppercase" ;
6.	LTRIM( )	Removes leading spaces	SELECT LTRIM(' RDBMS MySQL') ;
7.	RTRIM( )	Removes trailing spaces	SELECT RTRIM(' RDBMS MySQL ') ;
8.	TRIM( )	Removes leading and trailing spaces	SELECT TRIM(' Bar One ') ;
9.	INSTR( )	Returns the index of the first occurrence of substring	SELECT INSTR('CORPORATE FLOOR', 'OR') AS Instring ;
10.	LENGTH( )	Returns the length of a string in bytes	SELECT LENGTH('CANDIDE') "Length in characters" ;
11.	LEFT( )	Returns the leftmost number of characters as specified	SELECT LEFT('USS/23/67/09', 3) ;
12.	RIGHT( )	Returns the specified right-most number of characters	SELECT RIGHT('USS/23/67/09', 2) ;
13.	MID( )	Returns a substring starting from the specified position	SELECT SUBSTRING('Quadratically',5,6); or SELECT MID('Quadratically',5,6);

13.12.2 Numeric Functions

The number functions are those functions that accept numeric values and after performing the required operation, return numeric values. Some useful numeric functions are being discussed below :

	Function	Description	Example
1.	MOD()	Returns the remainder of one expression by diving by another expression.	SELECT MOD(11, 4) "Modulus" ;

	Function	Description	Example
2.	POWER()/POW()	Returns the value of one expression raised to the power of another expression	SELECT POWER(3, 2) "Raised" ;
3.	ROUND()	Returns numeric expression rounded to an integer. Can be used to round an expression to a number of decimal points	SELECT ROUND(15.193, 1) "Round" ;
4.	SIGN()	This function returns sign of a given number	SELECT SIGN(-15) "Sign" ;
5.	SQRT()	Returns the non-negative square root of numeric expression.	SELECT SQRT(25) "Square root" ;
6.	TRUNCATE()	Returns numeric exp1 truncated to exp2 decimal places. If exp2 is 0, then the result will have no decimal point.	SELECT TRUNCATE(15.79, 1) "Truncate" ;

13.12.3 Date and Time Functions

Date functions operate on values of the DATE datatype.

	Function	Description	Example
1.	CURDATE( )/ CURRENT_DATE( )/ CURRENT_DATE	Returns the current date	SELECT CURDATE( ) ;
2.	DATE( )	Extracts the date part of a date or datetime expression	SELECT DATE('2013-12-31 01:02:03') ;
3.	MONTH( )	Returns the month from the date passed	SELECT MONTH('2014-02-03') ;
4.	YEAR( )	Returns the year	SELECT YEAR('2014-02-03') ;
5.	DAYNAME( )	Returns the name of the weekday	SELECT DAYNAME('2014-02-03') ;
6.	DAYOFMONTH( )	Returns the day of the month (1-31)	SELECT DAYOFMONTH('2014-02-03') ;
7.	DAYOFWEEK( )	Returns the weekday index of the argument	SELECT DAYOFWEEK('2014-02-13') ;
8.	DAYOFYEAR( )	Return the day of the year (1-366)	SELECT DAYOFYEAR('2014-02-13') ;
9.	NOW( )	Returns the current date and time	SELECT NOW( ) ;
10.	SYSDATE( )	Returns the time at which the function executes	SELECT NOW(), SLEEP(2), NOW();

Let Us Revise

- *A function is a special type of predefined command set that performs some operation and returns a single value.*
- *Major Character functions are* **Lower/LCASE( ), Upper/UCASE( ), Concat( ), Instr( ), Length( ), RTrim( ), LTrim( ), Substr( ).**
- *Major Number functions are* **Round( ), Truncate( ), Mod( ), Sign( ).**
- *Major Date functions are* **Curdate( ), Date( ), Month( ), Year( ), DayName( ), DayofMonth( ), DayofWeek( ), DayofYear( ), Now( ), SysDate( ).**

13.13 Creating Tables with SQL Constraints

You already know that a table is created using CREATE TABLE command. The SQL syntax for **CREATE TABLE** is

```
CREATE TABLE "table_name"
("column 1" "data_type_for_column_1",
"column 2" "data_type_for_column_2", ... ) ;
```

So, if we are to create the customer table specified as above, we would type in

```
CREATE TABLE customer
(    First_Name char(50),
     Last_Name char(50),
     Address char(50),
     City char(50),
     Country char(25),
     Birth_Date date    ) ;
```

13.13.1 SQL Constraints

Common types of constraints include the following :

S.No.	Constraints	Description
1.	*NOT NULL*	❐ Ensures that a column cannot have NULL value.
2.	*DEFAULT*	❐ Provides a default value for a column when none is specified.
3.	*UNIQUE*	❐ Ensures that all values in a column are different.
4.	*CHECK*	❐ Makes sure that all values in a column satisfy certain criteria.
5.	*Primary Key*	❐ Used to uniquely identify a row in the table.
6.	*Foreign Key*	❐ Used to ensure referential integrity of the data.

13.13.1A SQL NOT NULL Constraint

By default, a column can hold NULL. If you not want to allow NULL value in a column, you will want to place a constraint on this column specifying that NULL is now not an allowable value.

DEF

A Constraint is a condition or check applicable on a field or set of fields.

For example, in the following statement,

```
CREATE TABLE Customer
(    SID integer NOT NULL,
     Last_Name varchar (30) NOT NULL,
     First_Name varchar(30)   );
```

Columns ***SID*** and ***Last_Name*** cannot include NULL, while ***First_Name*** can include NULL.

An attempt to execute the following SQL statement,

```
INSERT INTO Customer (Last_Name, First_Name)
VALUES ('Wang','Pedro');
```

will result in an error because this will lead to column *SID* being NULL, which violates the NOT NULL constraint on that column.

13.13.1B SQL DEFAULT Constraint

The DEFAULT constraint provides a *default value* to a column when the INSERT INTO statement does not provide a specific value. For example, if we create a table as below :

```
CREATE TABLE Student
(    Student_ID integer Unique,
     Last_Name varchar (30),
     First_Name varchar (30),
     Score DEFAULT 80 );
```

and execute the following SQL statement,

```
INSERT INTO Student (Student_ID, Last_Name, First_Name)
VALUES ('10','Qureshi','Zeeshan') ;
```

See, no value has been provided for Score field

After this SQL query, the table will look like the following :

Student_ID	Last_Name	First_Name	Score
10	Qureshi	Zeeshan	80

See, Score field has got the default value

Even though we didn't specify a value for the ***Score*** column in the INSERT INTO statement, it does get assigned the *default value* of *80* since we had already set *80* as the *default value* for this column.

13.13.1C SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are distinct. In other words, no two rows can hold the same value for a column with UNIQUE constraint.

For example, in the following CREATE TABLE statement,

```
CREATE TABLE Customer
(    SID integer Unique,
     Last_Name varchar (30),
     First_Name varchar(30)  );
```

Column ***SID*** has a unique constraint, and hence cannot include duplicate values. Such constraint does not hold for columns ***Last_Name*** and ***First_Name***. So, if the table already contains the following rows :

SID	Last_Name	First_Name
1	Wang	Pedro
2	Qureshi	Zeeshan
3	Rastogi	Rajiv

Executing the following SQL statement,

```
INSERT INTO Customer
VALUES ('3','Cyrus','Grace');
```

will result in an error because the value 3 already exists in the *SID* column, thus trying to insert another row with that value violates the UNIQUE constraint.

NOTE

Please note that a column that is specified as a primary key must also be unique. At the same time, a column that's unique may or may not be a primary key. In addition, multiple UNIQUE constraints can be defined on a table.

13.13.1D SQL CHECK Constraint

The CHECK constraint ensures that all values in a column satisfy certain conditions. Once defined, the database will only insert a new row or update an existing row if the new value satisfies the CHECK constraint. The CHECK constraint is used to ensure data quality

For example, in the following CREATE TABLE statement,

```
CREATE TABLE Customer
(    SID integer CHECK (SID > 0),
     Last_Name varchar (30),
     First_Name varchar(30) );
```

Column *SID* has a constraint — *its value must only include integers greater than 0*. So, attempting to execute the following statement,

```
INSERT INTO Customer VALUES ('-3','Gonzalves','Linda');
```

should result in an error because the values for SID must be greater than 0. **But you will find that your query is executed without checking the condition specified through CHECK constraint.** *The reason being* MySQL ignores CHECK constraint internally.

NOTE

Please note that MySQL does not enforce the CHECK constraint although MySQL syntax allows the definition of CHECK constraint. As per MySQL documentation, 'Check constraint gets parsed but IGNORED'.

13.13.1E PRIMARY KEY Constraint

A primary key is used to uniquely identify each row in a table. It can either be part of the actual record itself, or it can be an artificial field (one that has nothing to do with the actual record). A primary key can consist of one or more fields on a table. When multiple fields are used as a primary key, they are called a **composite key**.

Primary keys can be specified either when the table is created (using **CREATE TABLE**) or by changing the existing table structure (using **ALTER TABLE**).

Defining Primary Key through Create Table command

You can define a primary key in CREATE TABLE command through keywords PRIMARY KEY.

Below are examples for specifying a primary key when creating a table :

```
CREATE TABLE Customer
(    SID integer not null PRIMARY KEY,
     Last_Name varchar(30),
     First_Name varchar(30)  );
```

Or

```
CREATE TABLE Customer
(    SID integer not null,
     Last_Name varchar(30),
     First_Name varchar(30),
     PRIMARY KEY (SID)  );
```

The latter way is useful if you want to specify a composite primary key *e.g.*,

```
CREATE TABLE Customer
(    Branch integer not null,
     SID integer not null,
     Last_Name varchar(30),
     First_Name varchar(30),
     PRIMARY KEY (Branch, SID) );
```

*Combination of fields **Branch and SID** is the composite primary key of the table.*

Defining Primary Key through Alter Table command

You can define a primary key in ALTER TABLE command through keywords

```
ADD PRIMARY KEY (<key-field>)
```

Below are examples for specifying a primary key by altering a table :

```
ALTER TABLE Customer
ADD PRIMARY KEY (SID);
```

NOTE

Before using the ALTER TABLE command to add a primary key, you'll need to make sure that the field is defined as 'NOT NULL' – in other words, NULL cannot be an accepted value for that field.

13.13.1F FOREIGN KEY Constraint

In an RDBMS, tables reference one another through common fields and to ensure validity of references, *referential integrity* is enforced. *Referential integrity is a system of rules that a DBMS uses to ensure that relationships between records in related tables are valid, and that users don't accidentally delete or change related data.* Referential integrity is ensured through FOREIGN KEY constraint. This is implemented as explained in the following paragraph.

Whenever two tables are related by a common column (or set of columns), then the related column(s) in the **parent table** (or **primary table**) should be either declared a PRIMARY KEY or UNIQUE key and the related column(s) in the **child table** (or **related table**) should have FOREIGN KEY constraint.

For instance, if we have two tables having structures as given below :

Table : CUSTOMER

column name	characteristic
SID	Primary Key
Last_Name	
First_Name	

Table : ORDERS

column name	characteristic
Order_ID	Primary Key
Order_Date	
Customer_SID	Foreign Key
Amount	

In the above example, the **Customer_SID** column in the *ORDERS* table is a foreign key pointing to the SID column in the *CUSTOMER* table.

Just like primary key, Foreign key can also be created in *two* ways : through CREATE TABLE and ALTER TABLE commands.

Defining Foreign key through Create Table

In Create Table command, you can add foreign key's definition through following syntax :

```
Foreign Key (<column-to-be-designated-as-foreign-key>) references
Master-Table(<primary-key-of master-table>);
```

Following example shows how to specify the foreign key when creating the ORDERS table :

```
CREATE TABLE ORDERS
(    Order_ID integer,
     Order_Date date,
     Customer_SID integer,
     Amount double,
     Primary Key (Order_ID),
     Foreign Key (Customer_SID) references CUSTOMER(SID)  );
```

The above code will designate **Customer_SID** field of *ORDERS* table as foreign key referencing **SID** field of *CUSTOMER* table.

Defining Foreign key through Alter Table

In Alter Table command, you can add foreign key's definition through following syntax :

```
ALTER TABLE <table-name>
ADD FOREIGN KEY (<column-to-be-designated-as-foreign-key>)
references Master-Table(<primary-key-of master-table>) ;
```

Following example specifies a foreign key by altering a table. This assumes that the ORDERS table has been created, and the foreign key has not yet been put in :

```
ALTER TABLE ORDERS
ADD FOREIGN KEY (customer_sid) REFERENCES CUSTOMER(SID);
```

IMPORTANT

Foreign Key and Storage Engine

MySQL is capable of creating databases in many different storage engines that offer different features. The default storage engine is ISAM but ironically it does not support foreign keys. So, if you find that your foreign key constraint are being ignored or not implemented, you need to change the storage engine of your database tables to InnoDB storage engine.

To do so you simply need to write following command for your connecting tables :

```
ALTER TABLE <tablename> ENGINE = innodb ;
```

e.g.,

```
ALTER TABLE table1 ENGINE = innodb ;
```

Once done, you can create and implement foreign keys for your tables.

13.13.2 Applying Table Constraints

When a constraint is to be applied on a group of columns of the table, it is called ***table constraint***. The table constraints appear in the end of table definition. For instance, if you want combination of *icode* and *descp* of table *items* to be unique, you may write it as follows :

```
CREATE TABLE items
(   icode     char (5)      NOT NULL,
    descp     char (20)     NOT NULL,
    ROL       integer,
    QOH       integer,
    CHECK     (ROL < QOH),
    UNIQUE    (icode, descp) ) ;
```

these are a table constraints

The above statement ensures that the combination of *icode* and *desc* in each row must be unique.

A constraint applied on one column (*e.g.*, as you define not null with a column definition) is known as **column constraints.**

13.14 Viewing a Table Structure

Once you have created a table, you may want to view its structure. Anytime, if you want to view an already created/existing table's structure, you may use DESC[RIBE] command of MySQL.

You can use DESC[RIBE] command as follows :

```
DESC[RIBE] <tablename> ;
```

For example,

```
DESC empl ;
```

or

```
DESCRIBE empl ;
```

13.15 Inserting Data Into Table

You already know that data is added to tables using INSERT INTO command *e.g.,*

```
INSERT INTO employee
VALUES (1001, 'Rahman', 'M', 'E4', 14700.00) ;
```

Or

```
INSERT INTO employee (ecode, ename, sex)
VALUES (3005, 'Meera', 'F') ;
```

Inserting Data from Another Table

INSERT command can also be used to take or derive values from one table and place them in another by using it with a query. To do this, simply replace the VALUES clause with an appropriate query as shown in the following example :

```
INSERT INTO branch1
SELECT * FROM branch2
WHERE gross > 7000.00 ;
```

This query will generate data to be inserted into table

It will extract all those rows from *branch2* that have *gross* more than 7000.00 and insert this produced result into the table *branch1.*

13.16 Modifying Data in Tables

You can modify data in tables using UPDATE command of SQL. The UPDATE command specifies the rows to be changed using the WHERE clause, and the new data using the SET keyword. The new data can be a specified constant, an expression or data from other tables. For example, to change the reorder level *ROL* of all items to 250, you would write

```
UPDATE items
SET ROL = 250 ;
```

To update the *ROL* and *QOH* for items having icode less than 'I040', we shall write

```
UPDATE items
SET ROL = 400, QOH = 700
WHERE icode < 'I040' ;
```

To double the gross pay of employees of grade 'E3' and 'E4', you use the command :

```
UPDATE employee
SET gross = gross * 2
WHERE (grade = 'E3' OR grade = 'E4') ;
```

13.17 Deleting Data from Tables

To delete some data from tables, you can use SQL DELETE commands. The DELETE command removes rows from a table. This removes the entire rows, not individual field values, so no field argument is needed or accepted.

The DELETE statement takes the following general form :

```
DELETE FROM <tablename>
[WHERE <predicate> ] ;
```

For instance, to remove the tuples from *employee* table that have *gross* (salary) less than 2200.00, the following command is used :

```
DELETE FROM employee
WHERE gross < 2200.00 ;
```

If you specify no condition with WHERE, then all the rows of the table will be deleted, *e.g.*,

To remove all the contents of *items* table, you use the command :

```
DELETE FROM items ;
```

13.18 Altering Tables

The ALTER TABLE command is used to change definitions of existing tables. Usually, it can add columns to a table. Sometimes it can delete columns (depending on privileges) or change their sizes. In general, in MySQL SQL, ALTER TABLE command is used :

- to add a column
- to add an integrity constraint
- to redefine a column (datatype, size, default value).

You can use ALTER TABLE **for adding columns** as follows :

```
ALTER TABLE <table name> ADD <column name> <data type><size> [<constraint name>] ;
```

For instance, to add a new column *tel_number* of type *integer* in table *Empl* you may give :

```
ALTER TABLE Empl
ADD (tel_number integer) ;
```

To *modify existing columns of table*, ALTER TABLE command can be used according to following syntax:

```
ALTER TABLE <tablename>
MODIFY (columname newdatatype (newsize) ) [FIRST|AFTER column] ;
```

To modify column *Job* of table *Empl* (*See below, Solved Problem 5 of this chapter*) to have new width of 30 characters, you may give :

```
ALTER TABLE Empl
MODIFY (Job char(30) ) ;
```

Sometimes you may need to **change the name of one of your columns**. For this you can use CHANGE clause of ALTER TABLE command as per following syntax :

```
ALTER TABLE
CHANGE [COLUMN] old_col_name new_col_name column_definition ;
```

For instance, to change the existing column namely *First_Name* of table *Student*, to *FirstName* you may write :

```
ALTER TABLE Customers
CHANGE First_Name  FirstName VARCHAR(20);
```

See, complete column description is given along with the new name.

13.19 Dropping Tables

The DROP TABLE command of SQL lets you drop a table from the database. The syntax for using a DROP TABLE command is :

```
DROP TABLE [IF EXISTS] <tablename>
```

That is, to drop a table items, you need to write :

```
DROP TABLE items ;
```

Once this command is given, the table name is no longer recognized and no more commands can be given on that object.

The IF EXISTS clause of DROP TABLE first checks whether the given table exists in the database or not. If it does, then it drops the mentioned table from the database. For instance, consider the following query :

```
DROP TABLE IF EXISTS players ;
```

The above query will first check for existence of *players* table in current database. If it exists, then it (table players) will be dropped from the database.

Let Us Revise

- *CREATE TABLE command is used to create tables in database.*
- *INSERT INTO command is used to insert data in the table.*
- *To insert data from other tables, subquery can be used inside INSERT INTO command.*
- *Existing data in tables can be changed with UPDATE command.*
- *Tuples in a table can be deleted using DELETE command.*
- *ALTER TABLE command is used to alter the definition of already created tables.*
- *With ALTER TABLE, new columns can be added, existing columns can be redefined.*
- *DROP TABLE command drops a table from a database.*

Let Us Explore

1. What are different limits of MySQL 5.1?

   Ans. 1. The maximum number of tables that can be referenced in a single join is 61.
   2. The number of tables that can be referenced in the definition of a view is 61.
   3. There is a hard limit of 4096 columns per table, but the effective maximum may be less for a given table.
   4. Every table has a maximum row size of 65,535 bytes. This maximum applies to all storage engines, but a given engine might have additional constraints that result in a lower effective maximum row size.

2. *On your MySQL server instance, there is a database Mydb that contains three tables namely class, marks, sports. To delete all the tables and database, following set of commands is issued, but it generates errors. What could be the problem.*

```
DROP DATABASE mydb ;
DROP TABLE class ;
DROP TABLE marks ;
DROP TABLE sports ;
```

Ans. The first statement drops the database. When a database is dropped, all its components such as tables etc also get dropped. Thus, with the first statement the tables also get dropped along with the database and no more exist. So the next three DROP TABLE statements are redundant and are not required.

Solved Problems

1 *What do you understand by MySQL server instance ?*

SOLUTION. MySQL server instance is created from background processes and applications. It stays in memory and listens for client requests coming in over the network and accesses database contents according to those requests and provides that to the clients.

2 *What do you understand by MySQL client ?*

SOLUTION. MySQL Clients are programs that connect to the MySQL server and issue queries in a pre-specified format. MySQL is compatible with the standards based SQL (Structured Query Language). The client program may contact the server programmatically or manually.

3 *What is SQL ? What are different categories of commands available in SQL ?*

SOLUTION. In order to access data within the Oracle database, all programs and users must use, Structured Query Language (SQL). SQL is the set of commands that is recognised by nearly all RDBMSs. SQL commands can be divided into following categories :

1. Data Definition Language (DDL) Commands.
2. Data Manipulation Language (DML) Commands.
3. Transaction Control Language (TCL) Commands.
4. Session Control Commands.
5. System Control Commands.

4 *Differentiate between DDL and DML commands.*

SOLUTION. The Data Definition Language (DDL) commands, as the name suggests, allow you to perform tasks related to data definition. That is, through these commands, you can perform tasks like, create, alter and drop schema objects, grant and revoke privileges etc.

The Data Manipulation Language (DML) commands, as the name suggests, are used to manipulate data. That is, DML commands query and manipulate data in existing schema objects.

5 *Differentiate between CHAR and VARCHAR datatypes.*

SOLUTION. The difference between CHAR and VARCHAR is that of *fixed length* and *variable length*. The CHAR datatype specifies a *fixed length* character string. When a column is given datatype as CHAR(*n*), then MySQL ensures that all values stored in that column have this length *i.e.*, *n* bytes. If a value is shorter than this length *n* then blanks are added, but the size of value remains *n* bytes.

VARCHAR, on the other hand, specifies a variable length string. When a column is given datatype as VARCHAR(*n*), then the maximum size a value in this column can have is *n* bytes. Each value that is stored in this column stores exactly as we specify it *i.e.*, no blanks are added if the length is shorter than maximum length *n*. However, if we exceed the maximum length *n*, then an error message is displayed.

6 *Which of the following operations is not database manipulation?*

(A) Retrieval (B) Modification (C) Loading a database into DBMS (D) Deletion of a record

SOLUTION. **(C)**

7 *What is the "data type"? What are main objectives of datatypes?*

SOLUTION. Data type is defined as a set of values along with the operations that can be performed on those values. Some common data types are *Integer, Float, Varchar, Char, String*, etc.

Main objectives of datatypes are :

- Optimum usage of storage space
- Represent all possible values
- Improve data integrity

Consider a table Empl as given below :

empno	ename	job	mgr	hiredate	sal	comm	deptno
8369	SMITH	CLERK	8902	1990-12-18	800.00	NULL	20
8499	ANYA	SALESMAN	8698	1991-02-20	1600.00	300.00	30
8521	SETH	SALESMAN	8698	1991-02-22	1250.00	500.00	30
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985.00	NULL	20
8654	MOMIN	SALESMAN	8698	1991-09-28	1250.00	1400.00	30
8698	BINA	MANAGER	8839	1991-05-01	2850.00	NULL	30
8882	SHIAVNSH	MANAGER	8839	1991-06-09	2450.00	NULL	10
8888	SCOTT	ANALYST	8566	1992-12-09	3000.00	NULL	20
8839	AMIR	PRESIDENT	NULL	1991-11-18	5000.00	NULL	10
8844	KULDEEP	SALESMAN	8698	1991-09-08	1500.00	0.00	30
8886	ANOOP	CLERK	8888	1993-01-12	1100.00	NULL	20
8900	JATIN	CLERK	8698	1991-12-03	950.00	NULL	30
8902	FAKIR	ANALYST	8566	1991-12-03	3000.00	NULL	20
8934	MITA	CLERK	8882	1992-01-23	1300.00	NULL	10

You can create above table by running the script file crEmpDep under SampleDB folder. For this, you need to simply write SOURCE D:/SamplDB/crEmpDep.sql *on* **mysql>** *prompt*

8 *Consider the sample menagerie database and answer the following queries :*

1. `SELECT DISTINCT type FROM event WHERE remark IS NULL;`
2. `SELECT * FROM pet WHERE owner LIKE "%e%" ;`
3. `SELECT name, sex FROM pet WHERE birth >= '1995-01-01';`

SOUTION.

1.

```
+-----------+
| type      |
+-----------+
| kennel    |
+-----------+
```

3.

```
+-----------+-----------+
| name      | sex       |
+-----------+-----------+
| Chirpy    | f         |
| Whistler  | NULL      |
| Slim      | m         |
| Puffball  | f         |
+-----------+-----------+
```

2.

```
+---------+---------+----------+---------+------------+------------+
| name    | owner   | species  | sex     | birth      | death      |
+---------+---------+----------+---------+------------+------------+
| Claws   | Gwen    | cat      | m       | 1994-03-17 | NULL       |
| Fang    | Benny   | dog      | m       | 1990-08-27 | NULL       |
| Bowser  | Diane   | dog      | m       | 1979-08-31 | 1995-07-29 |
| Chirpy  | Gwen    | bird     | f       | 1998-09-11 | NULL       |
| Whistler| Gwen    | bird     | NULL    | 1997-12-09 | NULL       |
| Slim    | Benny   | snake    | m       | 1996-04-29 | NULL       |
| Puffball| Diane   | hamster  | f       | 1999-03-30 | NULL       |
+---------+---------+----------+---------+------------+------------+
```

9 *Consider the sample menagerie database and answer the following queries :*

(*i*) `SELECT MONTH(birth), name FROM pet WHERE sex = 'm' ;`

(*ii*) `SELECT CONCAT ( CONCAT (species, '-'), sex ), name FROM pet ;`

(*iii*) `SELECT SUBSTR(name,1,3), species FROM pet WHERE sex = 'f' ;`

(*iv*) `SELECT YEAR( CURDATE() ) + YEAR(birth) FROM pet WHERE sex = 'f' ;`

SOLUTION.

(*i*)

month(birth)	name
3	Claws
8	Fang
8	Bowser
4	Slim

(*ii*)

concat (concat (species, '-'), sex)	name
cat-f	Fluffy
cat-m	Claws
dog-f	Buffy
dog-m	Fang
dog-m	Bowser
bird-f	Chirpy
NULL	Whistler
snake-m	Slim
hamster-f	Puffball

(*iii*)

substr(name,1,3)	species
Flu	cat
Buf	dog
Chi	bird
Puf	hamster

(*iv*)

year(curdate()) + year(birth)
4002
3998
4007
4008

10 *Predict the output of following queries :*

(*i*) `SELECT ROUND(29.21), ROUND(32.76);` (*ii*) `SELECT ROUND(29.21, 1), ROUND(32.76, 1);`

SOLUTION. (*i*) 29 33

(*ii*) 29.2 32.8

11 *Predict the output of following queries*

(*i*) `SELECT TRIM(TRAILING '.' FROM '  To be continued...');`

(*ii*) `SELECT TRIM('  To be continued...');`

SOLUTION. To be continued...

To be continued...

Contains leading spaces. Please note there are leading spaces before the word 'To'

12 *Predict the output of following queries :*

(*i*) `SELECT 9 mod 2;`

(*ii*) `SELECT CONCAT('Catch', 'a', 'falling', 'star');`

SOLUTION. (*i*) `1`

(*ii*) `Catchafallingstar`

13 (*a*) *Write one similarity and one difference between CHAR and VARCHAR data types.*

(*b*) *Consider the following table named "GARMENT". Write command of SQL for (i) to (iv) and output for (v) to (vii).*

Table : GARMENT

GCODE	GNAME	SIZE	COLOUR	PRICE
111	TShirt	XL	Red	1400.00
112	Jeans	L	Blue	1600.00
113	Skirt	M	Black	1100.00
114	Ladies Jacket	XL	Blue	4000.00
115	Trousers	L	Brown	1500.00
116	Ladies Top	L	Pink	1200.00

(*i*) *To display names of those garments that are available in 'XL' size.*

(*ii*) *To display codes and names of those garments that have their names starting with 'Ladies'.*

(*iii*) *To display garment names, codes and prices of those garments that have price in the range 1000.00 to 1500.00 (both 1000.00 and 1500.00 included).*

(*iv*) *To change the colour of garment with code as 116 to "Orange".*

(*v*) `SELECT COUNT (DISTINCT (SIZE)) FROM GARMENT :`

(*vi*) `SELECT AVG (PRICE) FROM GARMENT ;`

(*vii*) `SELECT GNAME FROM GARMENT WHERE SIZE IN ('M', 'L') AND PRICE > 1500 ;`

(*c*) *What is the degree and cardinality of 'Garment' table ?* **[CBSE D 15]**

SOLUTION. (*a*) Similarity between char and varchar datatype is both datatypes are used for Text/string type of data that can be of 1-255 characters in length.

Difference is, CHAR datatype specifies a fixed length character string whereas VARCHAR specifies a variable length character string.

(*b*) (*i*)
```
SELECT GNAME
FROM GARMENT
WHERE SIZE = 'XL'
```

(*ii*)
```
SELECT GCODE, GNAME
FROM GARMENT
WHERE GNAME LIKE 'Ladies%' ;
```

(*iii*)
```
SELECT GNAME, GCODE, PRICE
FROM GARMENT
WHERE PRICE BETWEEN 1000.00 AND 1500.00 ;
```

(*iv*)
```
UPDATE garment
SET COLOUR = 'orange'
WHERE GCODE = 116 ;
```

(*v*)
```
COUNT (DISTINCT (SIZE))
        3
```

(*vi*)
```
AVG(PRICE)
     1800.00
```

(*vii*)
```
GNAME
Jeans
```

(*c*)
```
degree - 5
cardinality - 6
```

14 *(a) Distinguish between Single Row and Aggregate functions of MySQL. Write one example of each.*

(b) Consider the following table named "SOFTDRINK". Write commands of SQL for (i) to (iv) and output for (v) to (vii).

Table : SOFTDRINK

DRINKCODE	DNAME	PRICE	CALORIES
101	Lime and Lemon	20.00	120
102	Apple Drink	18.00	120
103	Nature Nectar	15.00	115
104	Green Mango	15.00	140
105	Aam Panna	20.00	135
106	Mango Juice Bahaar	12.00	150

(i) To display names and drink codes of those drinks that have more than 120 calories.

(ii) To display drink codes, names and calories of all drinks, in descending order of calories.

(iii) To display names and price of drinks that have price in the range 12 to 18 (both 12 and 18 included).

(iv) Increase the price of all drinks in the given table by 10%.

(v) `SELECT COUNT (DISTINCT (PRICE)) FROM SOFTDRINK ;`

(vi) `SELECT MAX (CALORIES) FROM SOFTDRINK ;`

(vii) `SELECT DNAME FROM SOFTDRINK WHERE DNAME LIKE "%Mango%" ;`

(c) What is the degree and cardinality of "SOFTDRINK" TABLE ? [CBSE OD 15]

SOLUTION. (a) Single Row functions work with a single row at a time. A single row function returns a result for every row of a queried table. For example, pow( ), year( ), length( ) etc.

Aggregate functions work on a number of values of a column/expression and return a single value as a result. For example, MIN( ), MAX( ), SUM( ) etc.

(b) (i)
```
SELECT DNAME, DRINKCODE
FROM SOFTDRINK
WHERE CALORIES > 120 ;
```

(ii)
```
SELECT DRINKCODE, DNAME, CALORIES
FROM SOFTDRINK
ORDER BY CALORIES DESC ;
```

(iii)
```
SELECT DNAME, PRICE
FROM SOFTDRINK
WHERE PRICE BETWEEN 12 and 18 ;
```

(iv)
```
UPDATE SOFTDRINK
SET PRICE = PRICE + 0.10 * PRICE ;
```

(v)
```
COUNT (DISTINCT (PRICE))
      4
```

(vi)
```
MAX(CALORIES)
     150
```

(vii)
```
DNAME
Green Mango
Mango Juice Bahaar
```

(c)
```
degree – 4
cardinality – 6
```

15 *(a) Write two examples of DBMS software.* [CBSE D 2014]

(b) What is meant by NULL value in MySQL ?

(c) *Table 'Club' has 4 rows and 3 columns. Table 'Member' has 2 rows and 5 columns. What will be the cardinality of the Cartesian product of them ?*

(d) *A numeric data field CHANGER contains 25565.7765. Write commands to round off CHANGER*

(i) up to 2 decimal places (i.e., expected result 25565.78)

– *(ii) whole number (i.e., expected result 25566)*

SOLUTION. (a) (i) Oracle (ii) SQL Server (iii) MySQL (b) Null value signifies a legal empty value.

(c) 8

(d) (i)
```
SELECT ROUND(CHANGER, 2) ;
```
(ii)
```
SELECT Round(CHANGER) ;
```

16 *Write MySQL command to open an existing database.* **[CBSE OD 12]**

SOLUTION.
```
USE <databasename> ;
```
e.g., if database name is *Sample*, then the command to open this database would be :
```
USE Sample ;
```

17 *Ms. Mirana wants to remove the entire content of a table "BACKUP" alongwith its structure to release the storage space. What MySQL statement should she use ?* **[CBSE OD 12]**

SOLUTION. DROP TABLE BACKUP ;

18 *Mr. Janak is using a table with following columns :*

```
Name, Class, Course_Id, Course_name
```
[CBSE OD 12]

He needs to display names of students who have not been assigned any stream or have been assigned Course_name that ends with "economics".

He wrote the following command, which did not give the desired result.

```
SELECT Name, Class FROM Students
WHERE Course_name = NULL OR Course_name = "%economics" ;
```

Help Mr. Janak to run the query by removing the error and write the correct query.

SOLUTION.
```
SELECT Name, Class FROM Students
WHERE Course_name IS NULL OR Course_name LIKE "%economics" ;
```

19 (a) *Distinguish between ALTER TABLE and UPDATE commands of MySQL.* **[CBSE OD 2014]**

(b) *Mention two categories in which MySQL commands are broadly classified.*

(c) *Give two characteristics of Primary Key.*

(d) *A table FUNFOOD has 13 rows and 17 columns. What is the cardinality and degree of this table ?*

(e) *A numeric column MONEY contains 34567.7896. Write a command to truncate MONEY :*

(i) Up to 2 decimal places. (i.e., expected result 34567.78)

(ii) Up to 3 places. (i.e., expected result 34000)

SOLUTION.

(a) The **ALTER TABLE** is a DDL command that makes changes in the structure of the table. The **UPDATE** is a DML command that makes changes in the data stored in table.

(b) (i) DDL (Data Definition Language) (ii) DML (Data Manipulation Language)

(c) (i) It stores unique values for each row in the table. (ii) It cannot be empty or null.

(d) Cardinality : 13 ; Degree : 17

(e) (i)
```
SELECT truncate(MONEY, 2) ;
```
(ii)
```
SELECT truncate(MONEY, -3) ;
```

20 (a) *What is the purpose of GROUP BY clause in MySql ? How is it different from ORDER BY clause ?*

(b) *Table Hospital has 4 rows and 5 columns. What is the Cardinality and Degree of this table ?*

(c) *Consider the* ***Table Supplier*** *given below. Write command in MySql for (i) to (iv) and output for (v) to (vii)*

Table : Supplier

Scode	Pname	Supname	Qty	City	Price
101	Coffee	Nestle	200	Kolkata	55.00
102	Biscuit	Hide & Seek	100	Delhi	10.00
103	Jam	Kissan	110	Kolkata	25.00
104	Maggi	Nestle	150	Mumbai	10.00
105	Chocolate	Cadbury	170	Delhi	25.00
106	Sauce	Maggi	56	Mumbai	55.00
107	Cake	Britannia	72	Delhi	10.00

(i) *To display names of the products, whose Pname starts with 'B' in ascending order of Price ?*

(ii) *To display Supplier code, Product name and City of the products whose quantity is less than 150.*

(iii) *To count distinct City in the table.*

(iv) *To insert a new row in the table Supplier.*

'110', 'Bournvita', 'ABC', 170, 'Delhi', 40.00.

(v) *Select Pname from Supplier where Pname IN("Bread", "Maggi") ;*

(vi) *Select Count(distinct (City)) from Supplier ;*

(vii) *Select max(Price) form Supplier where City = "Kolkata" ;* **[CBSE D 12]**

SOLUTION.

(a) GROUP BY clause is used to create grouped summary results based on the value of a field. ORDER BY clause does not produce summary results, rather it orders the result in the order of a given field name.

(b) Cardinality = 4

Degree = 5

(c) (i)
```
SELECT Pname From Supplier
WHERE  Pname Like "B%"
ORDER BY Price ;
```

(ii)
```
SELECT Scode, Pname, City
FROM Supplier
WHERE Qty < 150 ;
```

(iii)
```
SELECT Count (distinct City)
FROM Supplier ;
```

(iv)
```
INSERT INTO Supplier
VALUES ("110", "Bournvita", "ABC", 170, "Delhi", 40.00) ;
```

(v) Maggi

(vi) 3

(vii) 55

21 *(a) State difference between date functions NOW( ) and SYSDATE( ) of MySql.*

(b) Name a function of MySQL which is used to remove trailing and leading spaces from a string.

(c) Consider the following table named "SBOP" with details of account holders. Write commands of MySql for (i) and (iv) and output for (v) to (vii).

Table : SBOP

Accountno.	Name	Balance	DateOfopen	Transaction
SB-1	Mr. Anil	15000.00	2011-02-24	7
SB-2	Mr. Amit	23567.89		8
SB-3	Mrs. Sakshi	45000.00	2012-02-04	5
SB-4	Mr. Gopal	23812.35	2013-09-22	
SB-5	Mr. Dennis	63459.80	2009-11-10	15

(i) To display Accountno, Name and DateOfopen of account holders having transactions more than 8.

(ii) To display all information of account holders whose transaction value is not mentioned.

(iii) To add another column Address with datatype and size as VARCHAR(25).

(iv) To display the month day with reference to DateOfopen for all the account holders.

(v)
```
SELECT Count(*) FROM SBOP ;
```

(vi)
```
SELECT Name, Balance FROM SBOP
WHERE Name LIKE "%i" ;
```

(vii)
```
SELECT ROUND (Balance, -3) FROM SBOP
WHERE AccountNo = "SB - 5" ;
```

[CBSE D 2014]

SOLUTION.

(a) The **NOW( )** function returns the current date and time at the start of statement execution.

The **SYSDATE( )** also returns current date but it returns the current time at which sysdate( ) executes, not the statement's start time.

e.g.,

```
SELECT now( ), sleep(3), sysdate( ), now( ) ;
```

The output produced by above statement, is :

now()	sleep(3)	sysdate()	now()
2014-04-26 15:52:48	0	2014-04-26 15:52:51	2014-04-26 15:52:48

Both now() functions return same time – the statement's start time

The sysdate() function returns different time here – the sysdate() function's start time

(b) Trim( )

(c) *(i)*
```
SELECT Accountno, Name, DateOfopen
FROM SBOP
WHERE Transaction > 8 ;
```

(ii)
```
SELECT * FROM SBOP
WHERE Transaction IS Null ;
```

(iii)
```
Alter Table SHOP
ADD Address VARCHAR(25) ;
```

(iv)
```
SELECT DAYOFMONTH(DateOfopen)
FROM SHOP ;
```

(v) 5

(vi) Mrs. Sakshi 45000.00

(vii) 63000

22 (a) *What is the difference between "%" and "_" wild card characters with reference to LIKE clause of MySQL ?*

(b) *Name a function of MySQL used to give the first occurrence of a string2 in string1.*

(c) *Consider the following table named "EXAM" with details of marks. Write command of MySQL for (i) to (iv) and output for (v) to (vii).*

Table : EXAM

Adno	SName	Percentage	Clsection	Stream
R001	Sushant	90.2	12A	Science
R002	Vaidyanath	80.5	12B	Humanities
R003	Mlara	68.9	12B	Science
R004	Nlara	96.0	12A	Commerce
R005	ShinJini	88.9	12D	Commerce

(i) *To display all information of the students of humanities in descending order of percentage.*

(ii) *To display Adno, Name, Percentage and Stream of those students whose name is less than 6 characters long.*

(iii) *To add another column Bus_Fees with datatype and size as Decimal (8, 2).*

(iv) *To increase percentage by 2% of all the Humanities students.*

(v)
```
SELECT COUNT (*) FROM Exam ;
```

(vi)
```
SELECT SName, Percentage FROM EXAM WHERE Name LIKE "N%" ;
```

(vii)
```
SELECT ROUND (Percentage, 0) FROM EXAM WHERE "R005" ;
```

[CBSE OD 2014]

SOLUTION.

(a) '%' wildcard replaces any number of characters whereas "_" wildcard replaces exactly one character *e.g.,*

```
LIKE "%A"
```

will give all values ending with A ; there can be as many characters before A.
But

```
LIKE "_A"
```

will return values having two characters and whose last character is A.

(b) INSTR( )

(c) (i)
```
SELECT * FROM EXAM
WHERE stream = "Humanties"
ORDER BY Percentage Desc ;
```

(ii)
```
SELECT Adno, SName, Percentage, Stream
FROM EXAM
WHERE Length(SName) < 6 ;
```

(iii)
```
Alter Table EXAM
Add Bus_Fees Decimal(8, 2) ;
```

(iv)
```
Update Exam
Set Percentage = Percentage + Percentage * 0.02
WHERE Stream = "Humanities" ;
```

(v) 5

(vi) Niara 96.0

(vii) 89

23 *What is the purpose of ALTER TABLE command in MySQL ? How is it different from UPDATE command ?* **[CBSE OD 11]**

SOLUTION. The ALTER TABLE command (a DDL command) changes a table structure in the database. It is used to add/change a column definition, add/change an integrity constraint definition etc.

The UPDATE command (a DML command) updates only the data rows in the table as per the condition specified in the WHERE clause. It does not bring any change to table structure.

24 *Consider the table EXAM given below. Write commands in MySql for (i) to (iv) and output for (v) to (vii)* **[CBSE OD 11]**

Table : EXAM

No.	Name	Stipend	Subject	Average	Division
1	Karan	400	English	68	FIRST
2	Aman	680	Mathematics	72	FIRST
3	Javed	500	Accounts	67	FIRST
4	Bishakh	200	Informatics	55	SECOND
5	Sugandha	400	History	35	THIRD
6	Suparna	550	Geography	45	THIRD

(i) *To list the names of those students, who have obtained Division as FIRST in the ascending order of NAME.*

(ii) *To display a report listing NAME, SUBJECT and Annual stipend received assuming that the stipend column has monthly stipend.*

(iii) *To count the number of students, who have either Accounts or Informatics as Subject.*

(iv) *To insert a new row in the table EXAM :*

6, "Mohan", 500, "English", 73, "SECOND" ;

(v) SELECT AVG(Stipend) FROM EXAM WHERE DIVISION = "THIRD" ;

(vi) SELECT COUNT (DISTINCT Subject) FROM EXAM;

(vii) SELECT MIN (Average) FROM EXAM WHERE Subject = "English";

SOLUTION. (i)
```
SELECT Name
FROM Exam
WHERE Division = "FIRST"
ORDER BY Name ;
```

(ii) SELECT Name, Subject, Stipend * 12 AS "Annual Stipend"
FROM Exam ;

(iii) SELECT Count(*)
FROM Exam
WHERE Subject IN ("Accounts", "Informatics") ;

(iv) INSERT INTO Exam VALUES (6, "Mohan", 500, "English", 73, "Second") ;

(v) 47.5

(vi) 6

(vii) 68

25 *Write MySql command will be used to open an already existing database "LIBRARY".* **[CBSE D 11]**

SOLUTION. USE LIBRARY ;

26 *The Mname Column of a table Members is given below :* **[CBSE D 11]**

Mname
Aakash
Hirav
Vinayak
Sheetal
Rajeev

Based on the information, find the output of the following queries :

(i) *Select Mname from members where mname like "%v";*

(ii) *Select Mname from members where mname like "%e%";*

SOLUTION. (i) Hirav
Rajeev

(ii) Sheetal
Rajeev

27 *Differentiate between Alternate key and Candidate key.* **[CBSE D 11 ; OD 11]**

SOLUTION.

Alternate key. A candidate key that is not serving as a primary key is called alternate key.

Candidate key. All possible combination of columns that can possibly serve as the primary key (i.e., can uniquely identify the rows in the table) are called candidate keys.

28 *Sarthya, a student of class XI, created a table "RESULT". Grade is one of the columns of this table. To find the details of students whose Grades have not been entered, he wrote the following MySql query, which did not give the desired result :*

```
SELECT * FROM Result WHERE Grade = "Null" ;
```

Help Sarthya to run the query by removing the errors from the query and write the correct Query. **[CBSE D 11]**

SOLUTION. **Grade = "NULL" is the error.**

Correct statement is :

```
SELECT * FROM Result
WHERE Grade is NULL ;
```

HOTS (Higher Order Thinking Skills)

SOLVED PROBLEMS

1. *You are creating the EMPLOYEES table. This table should contain the COMMISSION_PCT column and use a value of 10 percent if no commission value is provided when a record is inserted. Which line should you include in the CREATE TABLE statement to accomplish this task?*

   (i) `commission_pct NUMBER(4,2) DEFAULT 0.10`

   (ii) `commission_pct NUMBER(4,2) DEFAULT = 0.10`

   (iii) `commission_pct NUMBER(4,2) DEFAULT (0.10)`

   (iv) `commission_pct NUMBER(4,2) (DEFAULT, 0.10)`

   Solution *(i)* `commission_pct NUMBER(4,2) DEFAULT 0.10`

2. *You need to remove all the data in the SCHEDULE table, the structure of the table, and the indexes associated with the table. Which statement should you use?*

   (i) DROP TABLE *(ii)* TRUNCATE TABLE *(iii)* ALTER TABLE *(iv)* DELETE TABLE

   Solution. *(i)* DROP TABLE

3. *Examine the structure of the EMPLOYEES table:*

   ```
   EMPLOYEE_ID NUMBER Primary Key
   FIRST_NAME VARCHAR(25)
   LAST_NAME VARCHAR(25)
   ```

   Which three statements insert a row into the table? (Choose multiple)

   (A)
   ```
   INSERT INTO employees
   VALUES ( NULL, 'John','Smith');
   ```

   (B)
   ```
   INSERT INTO employees( first_name, last_name)
   VALUES('John','Smith');
   ```

   (C)
   ```
   INSERT INTO employees
   VALUES ('1000','John',NULL);
   ```

   (D)
   ```
   INSERT INTO employees(first_name,last_name, employee_id)
   VALUES ( 1000, 'John','Smith');
   ```

   (E)
   ```
   INSERT INTO employees (employee_id)
   VALUES (1000);
   ```

   (F)
   ```
   INSERT INTO employees (employee_id, first_name, last_name)
   VALUES ( 1000, 'John','');
   ```

   Solution. C, E, F.

   Since EMPLOYEE_ID column is used as primary key, it cannot be NULL, so only INSERT statements in C, E and F are correct. You can insert the row with NULL LAST_NAME as in Answer C, or only the row with EMPLOYEE_ID as in answer E, or the row with empty LAST_NAME column.

   Answer A is *incorrect*, because a primary key cannot be NULL.

   Answer B is *incorrect*, because INSERT statement does not contain primary key value at all, so this answer needs to be eliminated as correct one.

   Answer D is *incorrect*, because this statement shows incorrect order of columns of row which needs to be inserted into the table.

4. *You need to remove all the rows from the SALES_HIST table. You want to release the storage space, but do not want to remove the table structure. Which statement should you use?*

   (i) the DROP TABLE statement *(ii)* the ALTER TABLE statement

(iii) the DELETE statement

(iv) the TRUNCATE TABLE statement

Solution (iv) the TRUNCATE TABLE statement

5. *You added a PHONE_NUMBER column of NUMBER data type to an existing EMPLOYEES table. The EMPLOYEES table already contains records of 100 employees. Now, you want to enter the phone numbers of each of the 100 employees into the table. Some of the employees may not have a phone number available. Which data manipulation operation do you perform?*

A. MERGE B. INSERT C. UPDATE D. ADD E. ENTER

F. You cannot enter the phone numbers for the existing employee records.

Solution. C. To update information you need to use UPDATE command.

6. *On the EMPLOYEES table, EMPLOYEE_ID is the primary key. MGR_ID is the ID of managers and refers to the EMPLOYEE_ID. The JOB_ID column is a NOT NULL column.*

Evaluate this DELETE statement:

```
DELETE employee_id, salary, job_id
FROM employees WHERE dept_id = 90;
```

Why does the DELETE statement fail when you execute it?

Solution. We cannot specify column names in the DELETE clause of the DELETE statement.

UNSOLVED PROBLEMS

1. You need to display the last names of those employees who have the letter "A" as the second character in their names. Which SQL statement displays the required results ?

A.
```
SELECT last_name
FROM EMP
WHERE last_name LIKE '_A%';
```

B.
```
SELECT last_name
FROM EMP
WHERE last name = '*A%'
```

C.
```
SELECT last_name
FROM EMP
WHERE last name = '_A%';
```

D.
```
SELECT last_name
FROM EMP
WHERE last name LIKE '*A%'
```

2. Which four statements are true of the SQL statement given below:? (Choose four)

```
DROP TABLE DEPT;
```

A. You cannot roll back this statement.

B. All pending transactions are committed.

C. All views based on the DEPT table are deleted.

D. All indexes based on the DEPT table are dropped.

E. All data in the table is deleted, and the table structure is also deleted.

F. All data in the table is deleted, but the structure of the table is retained.

3. You own a table called EMPLOYEES with this table structure :

```
EMPLOYEE_ID   NUMBER          Primary Key
FIRST_NAME    VARCHAR(25)
LAST_NAME     VARCHAR(25)
HIRE_DATE     DATE
```

What happens when you execute this DELETE statement ?

```
DELETE employees;
```

A. You get an error because of a primary key violation.

B. The data and structure of the EMPLOYEES table are deleted.

C. The data in the EMPLOYEES table is deleted but not the structure.

D. You get an error because the statement is not syntactically correct.

4. Evaluate this SQL statement:

```
SELECT ename, sal, 12*sal+100 FROM emp ;
```

The SAL column stores the monthly salary of the employee. Which change must be made to the above syntax to calculate the annual compensation as "monthly salary plus a monthly bonus of Rs.1000, multiplied by 12" ?

A. No change is required to achieve the desired results.

B. `SELECT ename, sal, 12*(sal+1000) FROM emp ;`

C. `SELECT ename, sal, (12*sal)+1000 FROM emp ;`

D. `SELECT ename, sal+1000,*12 FROM emp ;`

Glossary

Data Dictionary	A file containing metadata i.e., data about data.
DataType	Means to identify type of data and associated operations for handling it.
Keyword	Word having a special meaning.
Query	A command given to produce certain specified information from the database table(s).
Function	Predefined command set that carries out an operation and returns a single value.
DDL	*Data Description Language* Subset of SQL commands that are used to describe various entities of database.
DML	*Data Manipulation Language* Subset of SQL commands that are used to manipulate data in tables.

Assignments

TYPE A : VERY SHORT ANSWER QUESTIONS

1. What is MySQL server ? What is MySQL client ?
2. What is SQL ? What are the different categories of SQL commands ?
3. What does Data Dictionary consist of ?
4. Maximum how many characters can be stored in a (*i*) text literal (*ii*) numeric literal ?
5. What is a datatype ? Name some datatypes available in MySQL.
6. What is null value in MySQL database ? Can you use nulls in arithmetic expressions ?
7. Which keyword eliminates the redundant data from a query result ?
8. How would you display system date as the result of a query ?
9. How would you calculate 13*15 in SQL ?
10. Define a function.
11. What will be the output of following codes ?

(*a*)
```
mysql> SELECT CONCAT (CONCAT ('Inform', 'atics'), 'Practices');
```

(*b*)
```
mysql> SELECT LCASE ('INFORMATICS PRACTICES CLASS 11TH') ;
```

(*c*)
```
mysql> SELECT UCASE ('Computer studies') ;
```

(*d*)
```
mysql> SELECT CONCAT (LOWER ('Class'), UPPER ('xii') ) ;
```

12. (*a*) What is MySQL ?

(*b*) Is NULL value the same as 0 (zero) ? Write the reason for your answer.

(*c*) Write the UPDATE command to increase the commission (Column name : COMM) by 500 of all the Salesmen who have achieved Sales (Column name : SALES) more than 200000. The table's name is COMPANY.

(*d*) While using SQL pattern matching, what is the difference between '_' (underscore) and '%' wildcard symbols ?

(*e*) Write one similarity and one difference between CHAR and VARCHAR data types. [CBSE OD 15]

13. Write SQL statement to display `Today, the date is <current date>`

14. (*a*) Write a command to add a NOT NULL constraint on FEES column of a student table.
(*b*) Write SQL command to create a SAVEPOINT called A1.
(*c*) Define Foreign Key with reference to RDBMS.
(*d*) Table BANK has 2 rows and 3 columns. Table CUSTOMER has 4 rows and 3 columns. What will be the cardinality and degree of the Cartesian product of them ?
(*e*) There is a column HOBBY in a Table CONTACTS. The following two statements are giving different outputs. What may be the possible reason ?

```
SELECT COUNT (*) FROM CONTACTS ;
SELECT COUNT (HOBBY) FROM CONTACTS ;
```

[CBSE D 2013]

15. (*a*) Sharmila wants to make the database named 'COMPANY' active and display the names of all the tables in it. Write MySQL commands for it.
(*b*) Write SQL command to remove column named 'Hobbies' from a table named 'Student'.
(*c*) Rewrite the following SQL statement after correcting error(s). Underline the corrections made :

```
INSERT IN EMP(EMPNO, SALES) VALUE (100, 20089.50) ;
```

(*d*) A table STUDENT has 5 rows and 3 columns. Table ACTIVITY has 4 rows and 2 columns. What will be the cardinality and degree of the Cartesian product of them ?
(*e*) Name the SQL commands used to :
(*i*) Physically delete a table from the database. (*ii*) Display the structure of a table. [CBSE D 15]

16. (*a*) Write a SQL command to view the constraints of EMP table.
(*b*) Mr. Krishnaswami is working on a database and has doubt about the concept of SAVEPOINT in a transaction. Write down the meaning of SAVEPOINT and provide a simple example considering yourself as an online web support excutive.
(*c*) What is the difference between CURDATE( ) and DATA( ) functions ?
(*d*) Table STUDENT has 4 rows and 2 columns. Table MARKS has 2 rows and 3 columns. What will be the cardinality and degree of the Cartesian product of STUDENT AND MARKS ?
(*e*) There is a column Salary in a Table EMPLOYEE. The following two statements are giving different outputs. What may be the possible reason ?

```
SELECT COUNT (*) FROM EMPLOYEE;
SELECT COUNT (Salary) FROM EMPLOYEE ;
```

[CBSE OD 2013]

17. What is a constraint ? Name some constraints that you can apply to enhance database integrity.

18. How is Primary key constraint different from Unique key constraint ? [CBSE OD 15]

Or

Write one similarity and one difference between UNIQUE and PRIMARY KEY constraints. [CBSE D 15]

19. What is primary key ? What is PRIMARY KEY constraint ?

20. What is NOT NULL constraint ? What is DEFAULT constraint ?

21. When a column's value is skipped in an INSERT command, which value is inserted in the database ?

22. What is the error in following statement ?

```
UPDATE EMPL ;
```

23. Identify the error : `DELETE ALL FROM TABLE EMPL ;`

24. Write MySql command to display the list of existing databases. [CBSE D 12]

25. Mr. William wants to remove all the rows form Inventory table to release the storage space, but he does not want to remove the structure of the table. What MySql statement should he use ? [CBSE D 12]

26. (*a*) What is the purpose of ORDER BY clause in MySql ? How is it different from GROUP BY clause ?
(*b*) Table SCHOOL has 4 rows and 5 columns. What is the Cardinality and Degree of this table ?

27. Write MySql command will be used to open an already existing database "CONTACTS". [CBSE OD 1]

TYPE B : SHORT ANSWER QUESTIONS

1. What is the role of database server in database management system ? Give the key features of MySQL.
2. How are SQL commands classified ?
3. Differentiate between DDL and DML commands.
4. (*a*) What is the use of UPDATE statement in SQL ? How is it different from ALTER statement ?
   (*b*) Mr. Shankar created a table VEHICLE with 3 rows and 4 columns. He added 1 more row to it and deleted one column. What is the Cardinality and Degree of the Table VEHICLE ?
   (*c*) Consider the following table named "GYM" with details about fitness items being sold in the store. Write command of SQL for (*i*) to (*iv*) and output for (*v*) to (*vii*).

Table : GYM

ICODE	INAME	PRICE	BRANDNAME
G101	Power Fit Exerciser	20000	Power Gymea
G102	Aquafit Hand Grip	1800	Reliable
G103	Cycle Bike	14000	Ecobike
G104	Protoner Extreme Gym	30000	Coscore
G105	Message Belt	5000	Message Expert
G106	Cross Trainer	13000	GTC Fitness

   (*i*) To display the names of all the items whose name starts with "A".
   (*ii*) To display ICODEs and INAMEs of all items, whose Brandname is Reliable or Coscore.
   (*iii*) To change the Brandname to "Fit Trend India" of the item, whose ICODE as "G101".
   (*iv*) Add a new row for new item in GYM with the details : "G107", "Vibro exerciser", 21000, "GTCFitness"
   (*v*) SELECT COUNT (DISTINCT (BRANDNAME)) FROM GYM ;
   (*vi*) SELECT MAX (PRICE) FROM GYM ;
   (*vii*) SELECT INAME FROM GYM WHERE INAME LIKE "%t" ; [CBSE D 2013]

5. (*a*) What is the use of COMMIT statement in SQL ? How is it different from ROLLBACK statement ?
   (*b*) Mr. James created a table **CLIENT** with 2 rows and 4 columns. He added 2 more rows to it and deleted one column. What is the Cardinality and Degree of the Table **CLIENT** ?
   (*c*) Consider the following table **FITNESS** with details about fitness products being sold in the store. Write command of SQL for (*i*) to (*iv*) and output for (*v*) to (*vii*)

Table : FITNESS

PCODE	PNAME	PRICE	Manufacturer
P1	Treadmill	21000	Coscore
P2	Bike	20000	Aone
P3	Cross Trainer	14000	Reliable
P4	Multi Gym	34000	Coscore
P5	Massage Chair	5500	Regrosene
P6	Belly Vibrator Belt	6500	Ambawya

   (*i*) To display the names of all the products with price more than 20000.
   (*ii*) To display the names of all products by the manufacturer "Aone".
   (*iii*) To change the price data of all the products by applying 25% discount reduction.

(iv) To add a new row for product with the details : "P7", "Vibro Exerciser", 28000, "Aone".

(v) SELECT * FROM FITNESS
WHERE MANUFACTURER NAME LIKE "%e" ;

(vi) SELECT COUNT(DISTINCT (MANUFACTURER)) FROM FITNESS ;

(vii) SELECT MAX (PRICE) FROM FITNESS ; [CBSE OD 2013]

6. Write SQL commands for the following on the basis of given table CLUB

Table : CLUB

COACH_ID	COACHNAME	AGE	SPORTS	DATOFAPP	PAY	SEX
1.	KUKREJA	35	KARATE	27/03/1996	1000	M
2.	RAVINA	34	KARATE	20/01/1998	1200	F
3.	KARAN	34	SQUASH	19/02/1998	2000	M
4.	TARUN	33	BASKETBALL	01/01/1998	1500	M
5.	ZUBIN	36	SWIMMING	12/01/1998	750	M
6.	KETAKI	36	SWIMMING	24/02/1998	800	F
7.	ANKITA	39	SQUASH	20/02/1998	2200	F
8.	ZAREEN	37	KARATE	22/02/1998	1100	F
9.	KUSH	41	SWIMMING	13/01/1998	900	M
10.	SHAILYA	37	BASKETBALL	19/02/1998	1700	M

(a) To show all information about the swimming coaches in the club.

(b) To list names of all coaches with their date of appointment (DATOFAPP) in descending order.

(c) To display a report, showing coachname, pay, age and bonus (15% of pay) for all the coaches.

(d) Give the output of following SQL statements :

(i) SELECT LCASE (SPORTS) FROM Club ;

(ii) SELECT MOD(Age, 5) FROM CLUB WHERE Sex = 'F' ;

(iii) SELECT POWER(3, 2) FROM CLUB WHERE Sports = 'KARATE' ;

(iv) SELECT SubStr(CoachName, 1, 2) FROM CLUB WHERE Datofapp > '1998-01-31' ;

7. Write SQL commands for the following on the basis of given table STUDENT

Table : STUDENT1

No.	Name	Stipend	Stream	AvgMark	Grade	Class
1	Karan	400.00	Medical	78.5	B	12B
2	Divakar	450.00	Commerce	89.2	A	11C
3	Divya	300.00	Commerce	68.6	C	12C
4	Arun	350.00	Humanities	73.1	B	12C
5	Sabina	500.00	Nonmedical	90.6	A	11A
6	John	400.00	Medical	75.4	B	12B
7	Robert	250.00	Humanities	64.4	C	11A
8	Rubina	450.00	Nonmedical	88.5	A	12A
9	Vikas	500.00	Nonmedical	92.0	A	12A
10	Mohan	300.00	Commerce	67.5	C	12C

(a) Select all the Nonmedical stream students from STUDENT1.

(*b*) List the names of those students who are in class 12 sorted by Stipend.

(*c*) List all students sorted by AvgMark in descending order

(*d*) Display a report, listing Name, Stipend, Stream and amount of stipend received in a year assuming that the Stipend is paid every month.

(*e*) Give the output of followint SQL statement :

(*i*) SELECT TRUNCATE(AvgMark) FROM Student1 WHERE AvgMark < 75 ;

(*ii*) SELECT ROUND(AvgMark) FROM Student1 WHERE Grade = 'B' ;

(*iii*) SELECT CONCAT(Name, Stream) FROM Student1 WHERE Class = '12A' ;

(*iv*) SELECT RIGHT(Stream, 2) FROM Student

8. What is foreign key ? How do you define a foreign key in your table ?
9. How is FOREIGN KEY commands different from PRIMARY KEY command ?
10. How is FOREIGN KEY commands related to the PRIMARY KEY ?
11. What are table constraints ? What are column constraints ? How are these *two* different ?
12. Insert all those records of table Accounts into table *Pending* where amt_outstanding is more than 10000.
13. Increase salary of employee recrods by 10% (*table employee*).
14. Add a constraint (NN-Grade) in table Empl that declares column Grade not null.
15. Drop the table Empl.
16. Differentiate between : (*i*) DROP TABLE, DROP DATABASE
(*ii*) DROP TABLE, DROP clause of ALTER TABLE.
17. Mr. Mittal is using a table with following columns : Name, Class, Stream_Id, Stream_name [CBSE D 12]
He needs to display names of students who have not been assigned any stream or have been assigned stream_name that ends with "computers".
He wrote the following command, which did not give the desired result.

```
SELECT Name, Class FROM Students
WHERE Stream_name = NULL OR Stream_name = "%computers" ;
```

Help Mr. Mittal to run the query by removing the error and write correct query.

18. Consider the Table SHOPPE given below. Write command in MySql for (*i*) to (*iv*) and output for (*v*) to (*vii*).

Table : SHOPPE

Code	Item	Company	Qty	City	Price
102	Biscuit	Hide & Seek	100	Delhi	10.00
103	Jam	Kissan	110	Kolkata	25.00
101	Coffee	Nestle	200	Kolkata	55.00
106	Sauce	Maggi	56	Mumbai	55.00
107	Cake	Britannia	72	Delhi	10.00
104	Maggi	Nestle	150	Mumbai	10.00
105	Chocolate	Cadbury	170	Delhi	25.00

(*i*) To display names of the items, whose name starts with 'C' in ascending order of Price ?

(*ii*) To display Code, Item name and City of the products whose quantity is less than 100.

(*iii*) To count distinct Company from the table.

(*iv*) To insert a new row in the table Shoppe.
'110', 'Pizza', 'Papa Jones', 120, 'Kolkata', 50.0.

(*v*) Select Item from Shoppe where Item IN("Jam", "Coffee") ;

(*vi*) Select Count(distinct (City)) from Shoppe ;

(*vii*) Select Min(Qty) form Shoppe where City = "Mumbai" ;

[CBSE OD 12]

19. The Doc_name Column of a table Hospital is given below : [CBSE OD 11]

Doc_nmae
Avinash
Hariharan
Vinayak
Deepak
Sanjeev

Based on the information, find the output of the following queries :

(*i*) SELECT doc_name FROM HOSPITAL WHERE Doc_name like "%v";

(*ii*) SELECT doc_name FROM HOSPITAL WHERE doc_name like "%e%";

20. Sarthak, a student of class XII, created a table "Class". Grade is one of the columns of this table. To find the details of students whose Grades have not been entered, he wrote the following MySql query, which did not give the desired result : [CBSE OD 11]

```
SELECT * FROM Class WHERE Grade = "Null";
```

Help Sarthak to run the query by removing the errors from the query and write the correct query.

21. What is the purpose of DROP TABLE command in MySql ? How is it different from DELETE command ? [CBSE D 11]

22. Consider the table RESULT given below. Write commands in MySql for (*i*) to (*iv*) and output for (*v*) to (*vii*) [CBSE D 11]

Table : RESULT

No	Name	Stipend	Subject	Average	Division
1	Sharon	400	English	38	THIRD
2	Amal	680	Mathematics	72	FIRST
3	Vedant	500	Accounts	67	FIRST
4	Shakeer	200	Informatics	55	SECOND
5	Anandha	400	History	85	FIRST
6	Upansna	550	Geography	45	THIRD

(*i*) To list the names of those students, who have obtained Division as FIRST in the ascending order of NAME.

(*ii*) To display a report listing NAME, SUBJECT and Annual stipend received assuming that the stipend column has monthly stipend.

(*iii*) To count the number of students, who have either Accounts or Informatics as Subject.

(*iv*) To insert a new row in the table EXAM :

6, "Mohan", 500, "English", 73, "Second"

(*v*) SELECT AVG(Stipend) FROM EXAM WHERE DIVISION = "THIRD"

(*vi*) SELECT COUNT (DISTINCT Subject) FROM EXAM;

(*vii*) SELECT MIN (Average) FROM EXAM WHERE Subject = "English";

TYPE C : LONG ANSWER QUESTION

1. What do you understand by client server architecture of MySQL ?