10

Relational Databases and Structured Query Language (SQL)

10.1 INTRODUCTION

When we speak about an organization, a large amount of data is required to be processed and handled. This data handling is performed by arranging data in the form of tables and databases.

A ***database*** is defined as an organized collection of data (information) about an entity (something that exists) or things. It is a shared collection of related data/information used to support the activities and decision-making of a particular organization. It also allows the users to enter, access and analyze their data quickly and easily. It serves as a container which may contain various database objects. Database is **integrated** as well as **shared**.

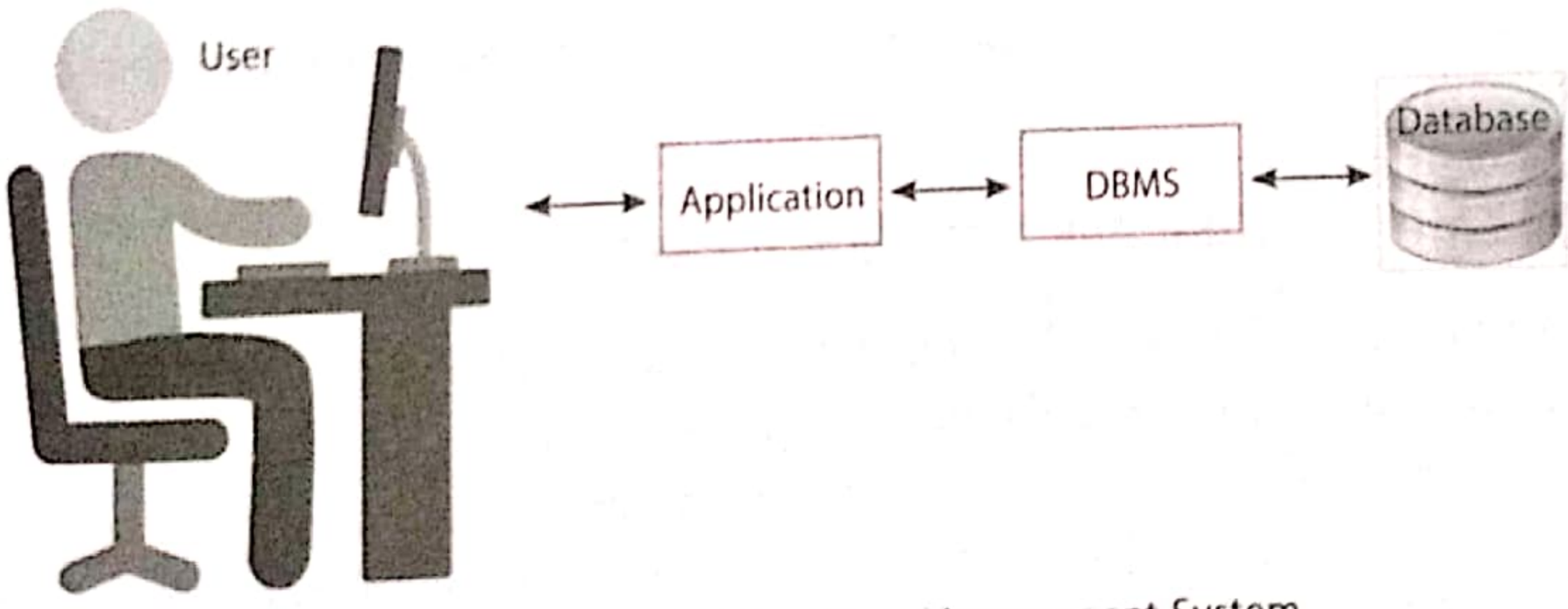


Fig. 10.1: Database and Database Management System

For example, all files belonging to one organization will be treated as the database of that organization. A database, therefore, is considered as the repository for the stored data. A few of the components that are an important part of a database like file or table, records, field, etc., will be discussed now.

CTM: Database is an organized collection of interrelated data that serves many applications.

Consider the example of a "**School**" database. This database shall constitute tables related to student, teacher, result, etc. The data is arranged inside a database as per the file organization hierarchy as shown in Fig. 10.2.

- A **field** is a set of characters which are used together to represent specific data elements. It is also termed as a data item. A specific or an individual data item within a record is known as a field.

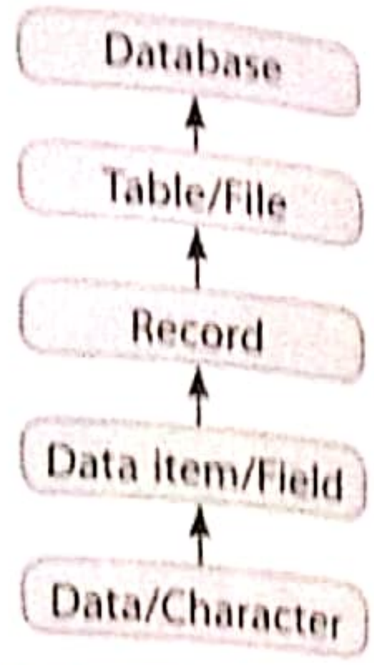


Fig. 10.2: File Organization

For example, roll number, name, age and marks are the fields in a student's record.

- A collection of fields is termed as **Record**. *For example*, a student record consists of the fields Roll No., Name, Age and Marks as shown in Fig.10.3.

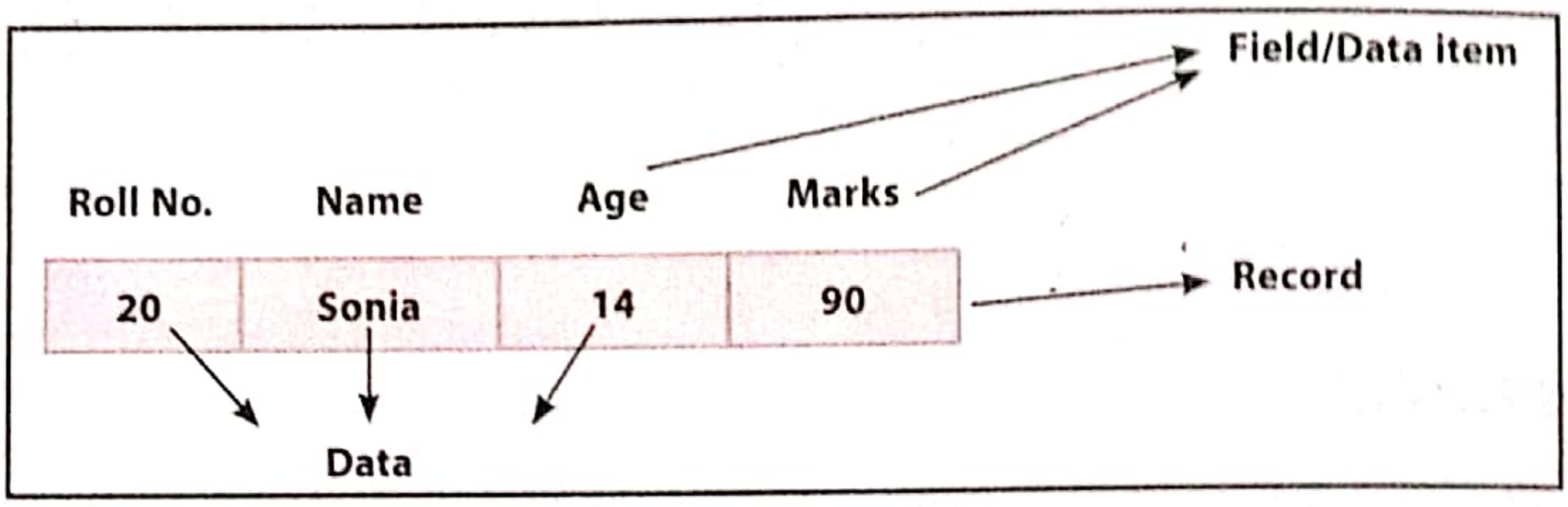


Fig. 10.3: Student table

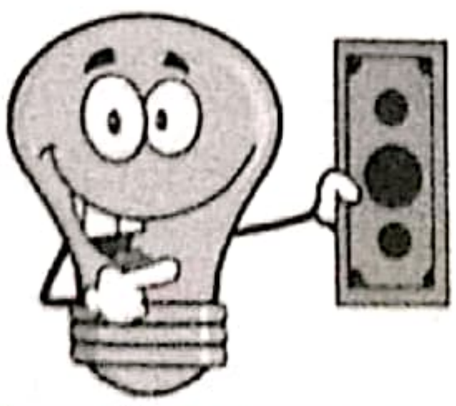
- A collection of logically related records is called a file. **A file is also termed as a table or a relation**. A table has rows and columns, where rows represent records or tuples and columns represent attributes or fields. *For example*, the entire information about all the students (in the form of records) in a class is kept in a file or table named "student" (Fig. 10.3).
- Database is, therefore, a place where related pieces of information are stored and various operations can be performed on them. It is the highest unit of file organization.

These databases are managed generally by special software known as ***Database Management System*** (***DBMS***).

10.2 DATABASE MANAGEMENT SYSTEM (DBMS)

Database Management System (DBMS) is not a new concept and was first implemented in the 1960s. With the passage of time, database technologies have evolved and usage and expected functionalities of databases have increased immensely. There is no organization which does not manage its data and records using any type of DBMS. An online telephone directory would definitely use DBMS to store data pertaining to people, phone numbers and other contact details. Apart from this, your electricity service provider uses a DBMS to manage billing, client-related issues, to handle fault data, etc. Facebook needs to store, manipulate and present data related to members, their friends, member activities, messages, advertisements and a lot more. All these real-life applications require some DBMS to manipulate and handle this enormous data. DBMS requires a language to handle and manipulate its data which is known as Structured Query Language (SQL).


Online Telephone Directory


Electricity Billing System

DBMSs are specially designed applications to create a connection between the user and the program, and to store data in an organized manner. The purpose of DBMS software is to allow the user to create, modify and control a database.

A DBMS stores data in such a manner that it becomes easier and highly efficient to retrieve, manipulate and produce information. Thus, a DBMS is an electronic or computerized record-keeping system which maintains the various pieces of information in an integrated and summarized form, instead of keeping them in separate independent files.

Examples of Database Management System are MS Access, MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP, dBase, FoxPro, etc.

A few customized DBMSs are computerized library systems, automated teller machines, flight reservation systems, computerized parts, inventory systems, etc.

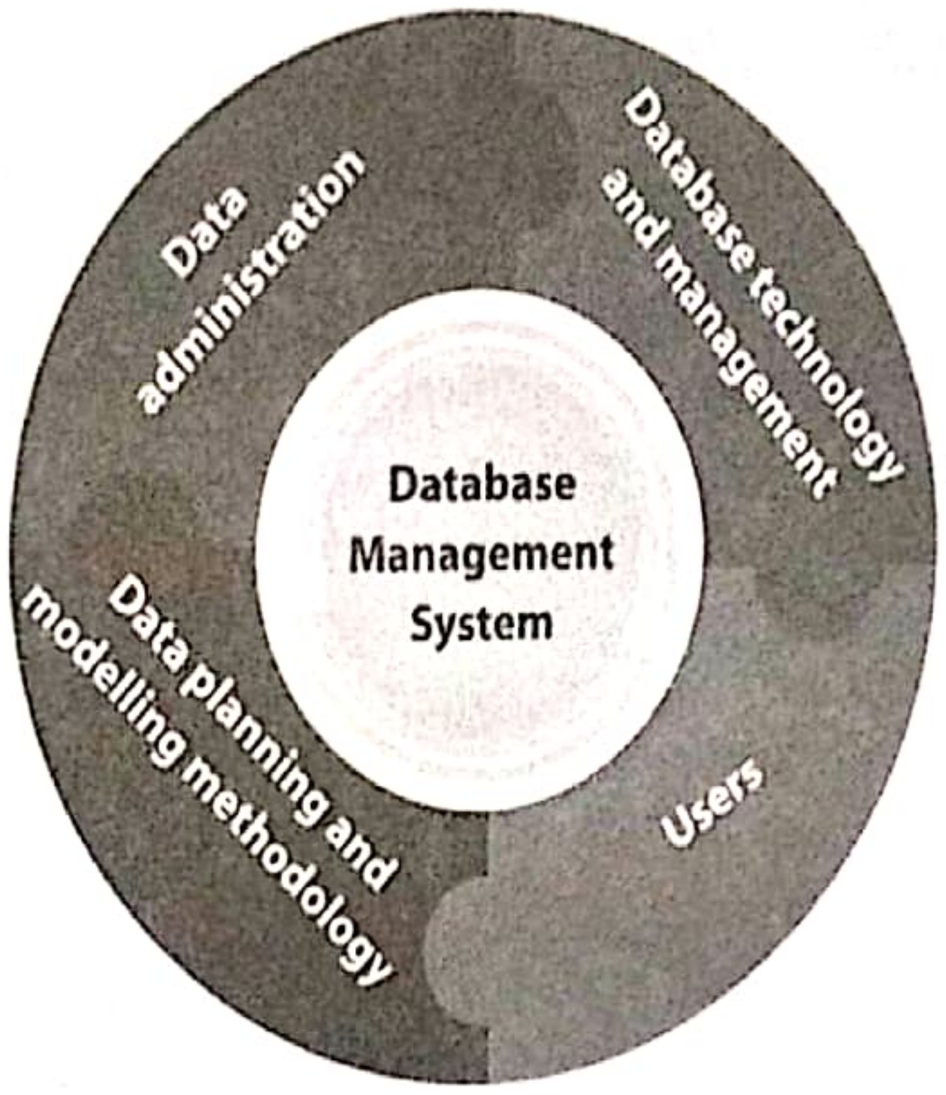


Fig. 10.4: Role of Database Management System

A DBMS gives us tools to:

1. store data in a structured way,
2. query the database (that is, ask questions about the data),
3. sort and manipulate the data in the database,
4. validate the data entered and check for inconsistencies,
5. produce flexible reports, both on screen and on paper, that make it easy to comprehend the information stored in the database.

CTM: A DBMS is a general purpose software system that facilitates the process of defining, constructing and manipulating databases for various applications.

Also, it maintains data consistency in case of multiple users.

10.2.1 Need for DBMS

The database system is used to eliminate the problems of data redundancy and data inconsistency. It does not maintain separate files for different applications. Rather, it works on the centrally maintained database which means that the data is kept at one place and all the applications that require the data may refer to this database. Whenever any file gets updated, the updated version of the file is available to all applications using the database system as shown in Fig. 10.5. So, data redundancy and data inconsistency are controlled to a large extent.

However, at times there might be data redundancy due to some technical requirements in business applications. In such cases, we are required to maintain the same data for different files, although this is not recommended.

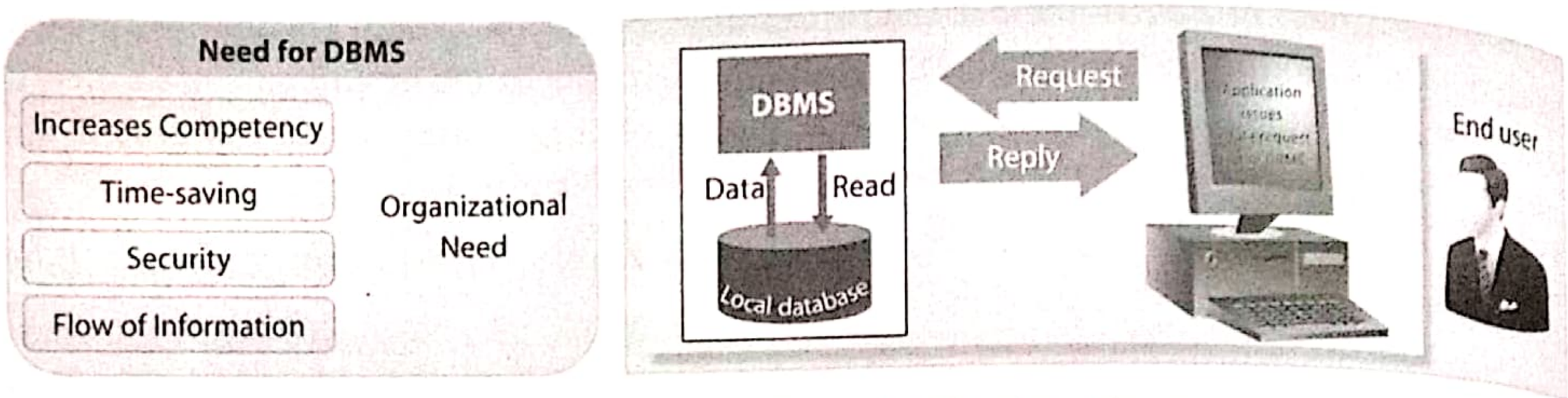


Fig. 10.5: Purpose of a Centralized Database System

10.2.2 Components of a Database System

The various components of a database system are described in the figure below:

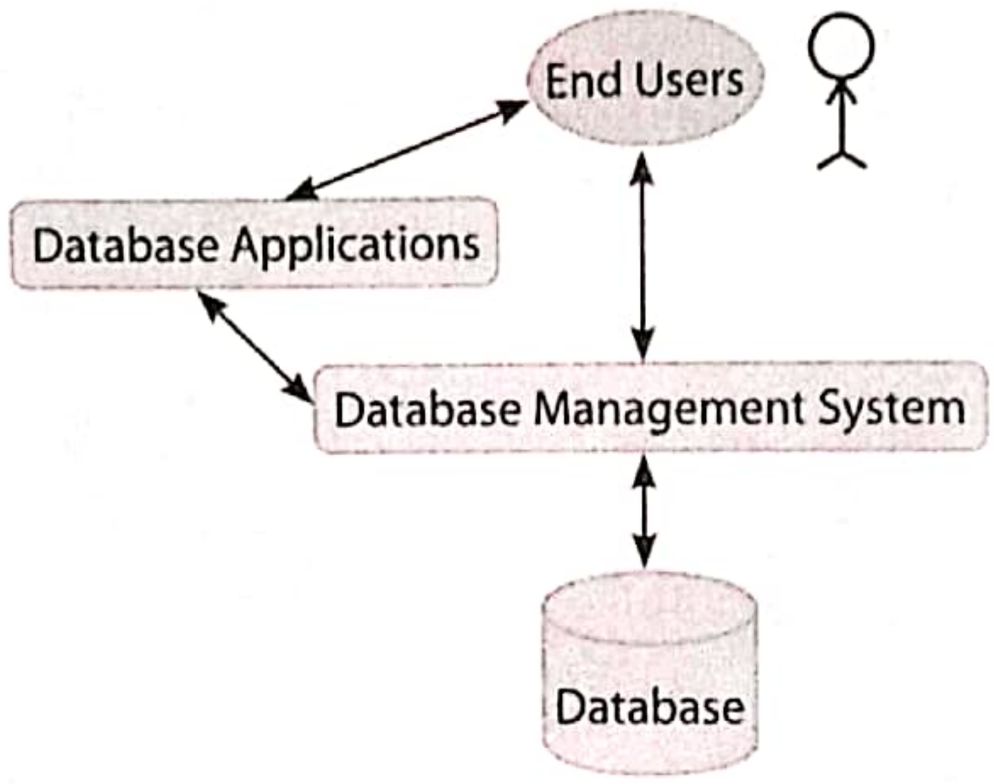


Fig. 10.6: Components of a Database System

Let us discuss these components.

1. **Users:** Users can be of varied types, usually a DB administrator, system or application developers and end-users. DBMS provides critical services to the user:

   (a) **Database Creation:** A DBMS helps the user in creating and defining the required data or, in turn, a database. It manages and organizes the required data and databases.

   (b) **Database Maintenance:** A DBMS provides maintenance of data and database by addition, deletion, modification and regular updation of the tables and its records.

   (c) **Database Processing:** A DBMS performs one of the major tasks of query processing– it processes the queries or the information requirement of the users and retrieves necessary information from the database.

2. **Database Application:** Database application may be Personal, Departmental, Operational and Internal. It may be general purpose or customized as per the needs of the user.
3. **DBMS:** Software that allows users to define, create, access and manage database(s) is termed as a DBMS. *For example*, MySQL, Oracle, etc.
4. **Database:** A collection of logical data.

10.2.3 Advantages of a DBMS

Apart from the various salient features described above, a DBMS has several advantages over traditional data processing techniques.

1. **Control of Data Redundancy:** Duplication of data leads to wastage of storage space. A DBMS eliminates data redundancy (duplication of data) by integrating the files so that multiple copies of the same data are not stored.
2. **Data Consistency:** DBMS provides data consistency to a large extent as the changes made at one place are reflected at all other places or to all the users.
3. **Sharing of Data:** Sharing of data using a DBMS implies that not only can the existing applications share the data in the database, but also that new applications can be developed to operate against the same stored data.
4. **Reduced Programming Effort:** A DBMS saves a lot of programming effort since a user need not write programs for query processing involving several tables or files, report generation, addition, modification and deletion of data, etc. Thus, it provides easy retrieval of data.
5. **Database Enforces Standards:** With centralized control of the database, the DBA (Database Administrator) can ensure that all applicable standards are followed in the representation of data, *i.e.*, format, documentation standards and conventions, etc.
6. **Improved Data Integrity:** Data integrity refers to the validity and consistency of stored data. *For example*, the system itself checks for the correct information to be entered by the user in the correct format. It consists of various constraints.
7. **Privacy and Security:** Data security refers to protection of data against accidental or intentional disclosure to unauthorized persons. Since there is a centralized control, the data is protected.
8. **Economical:** Combining all the organization's operational data into one database and creating a set of applications that works on this single source of data can result in cost-saving. The overall maintenance cost of data is reduced.
9. **Improved Backup and Recovery System:** A database system provides facilities for recovery from hardware or software failure.
10. **Provides Interface for Enterprise Requirement than Individual Requirement:** Since various types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interface.

CTM: The repetition (duplication) of the same data at multiple places in a database is known as **data redundancy**.

10.3 DBMS MODELS

Data models define how the logical structure of a database is modelled. A data model is an integrated collection of conceptual tools that can be used to describe the structure of the database along with its appropriate data types, relationships and constraints required to be applied on the data.

Data models are used to implement abstraction in a DBMS. Data models are a communication tool. These define how data is connected to each other and how they are processed and stored inside the system. Data models organize data for various users. A data model should be able to give the best data representation and should possess the following desirable characteristics:

1. Data models should be presented graphically using diagrams and symbols.
2. Data representation in a data model should have no data redundancy.
3. It should be made available and shared by various applications.
4. Data represented should be consistent, stable and valid in all aspects.

10.3.1 Types of Data Models

Data models are categorized into three different categories:

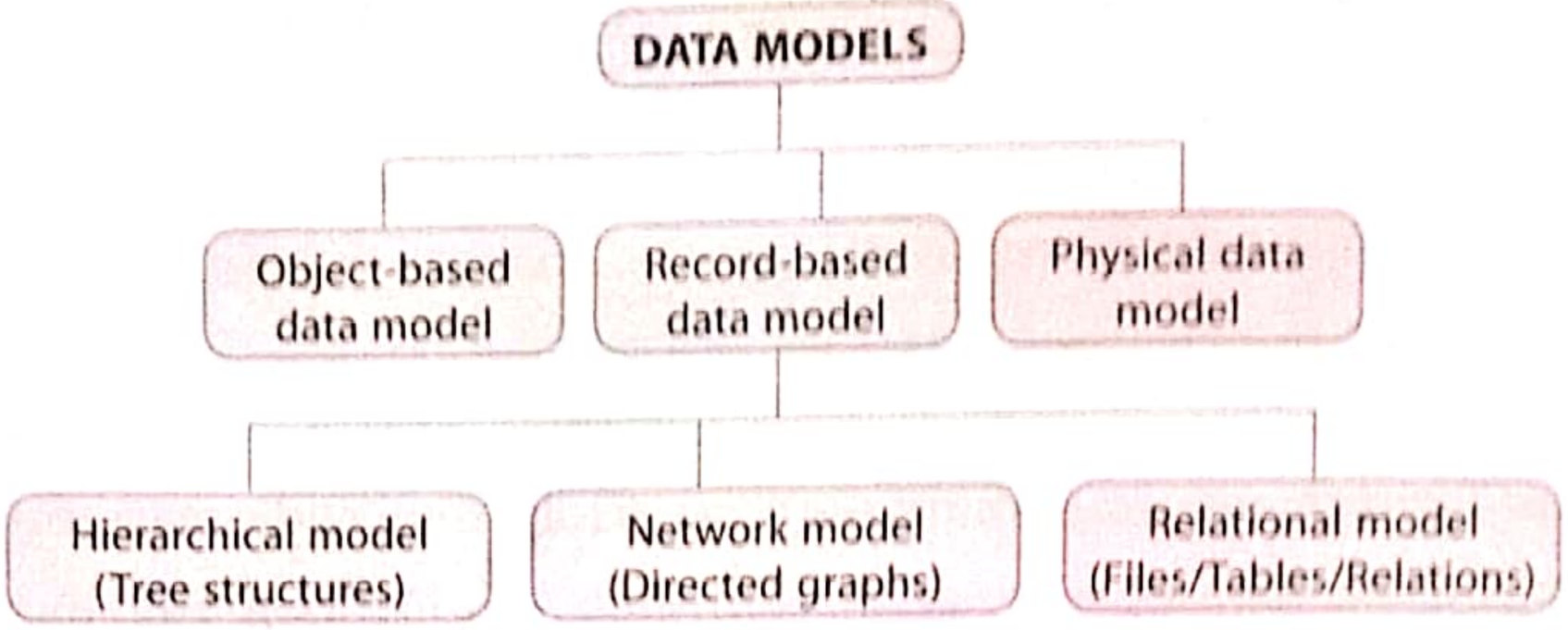


Fig. 10.7: Classification of Data Models

The above hierarchy describes the types of data models available. We will be talking only about Relational model as per the CBSE syllabus guidelines.

10.3.2 Relational Data Model

The most popular data model in DBMS is the Relational Model. It is a more scientific and stable model than hierarchical or network model since there are no pointers involved with the records and, in case of any fault or error, these pointers may result in inconsistency among the records and lead to reduced data integrity.

In this model, data is organized in two-dimensional tables called **relations**. The tables or relations are related to each other. A relational database is based on this relational model developed by E.F. Codd. In this type of database, the data and relations between them are organized into tables, having logically related records containing the same fields. The contents of a table or relation can be permanently saved for future use. Thus, this model is simple and has all the properties and capabilities required to process data with storage efficiency.

A relational model consists of a collection of tables, each of which is assigned a unique name, *i.e.*, it represents the database as a collection of relations or tables.

Characteristics of relational database are:

1. Data is conceptually represented as an orderly arrangement of data into rows and columns termed as a relation.
2. Each relation is represented as a table.
3. Columns described in a table are the attributes that belong to an entity which is modelled as a table.
4. Every row in a table represents a single entity.
5. All the values in a relation are scalar, *i.e.*, at any given row or column position, there is one and only one value.

Advantages of a relational model are as follows:

1. Relational model provides structural independence by using independent tables.
2. Changes made in the table structure do not affect data access or other application programs.
3. It is represented in the form of tables; so, it is simple and easy to understand.
4. Tabular view also provides simpler database design, use, implementation and management.
5. Built-in query support based on SQL is provided by RDBMS (Relational Database Management System).
6. Data organization and manipulation is easy, flexible and faster.
7. Powerful structure designing and processing capabilities of RDBMS isolate the end-user from physical-level details and, thus, improve implementation and management simplicity.
8. Mathematical operations can be successfully carried out using RDBMS.

Limitations of a relational model are:

1. The RDBMS incurs hardware and system software overheads.
2. The size of database becomes very large.

10.4 RELATIONAL DATABASE

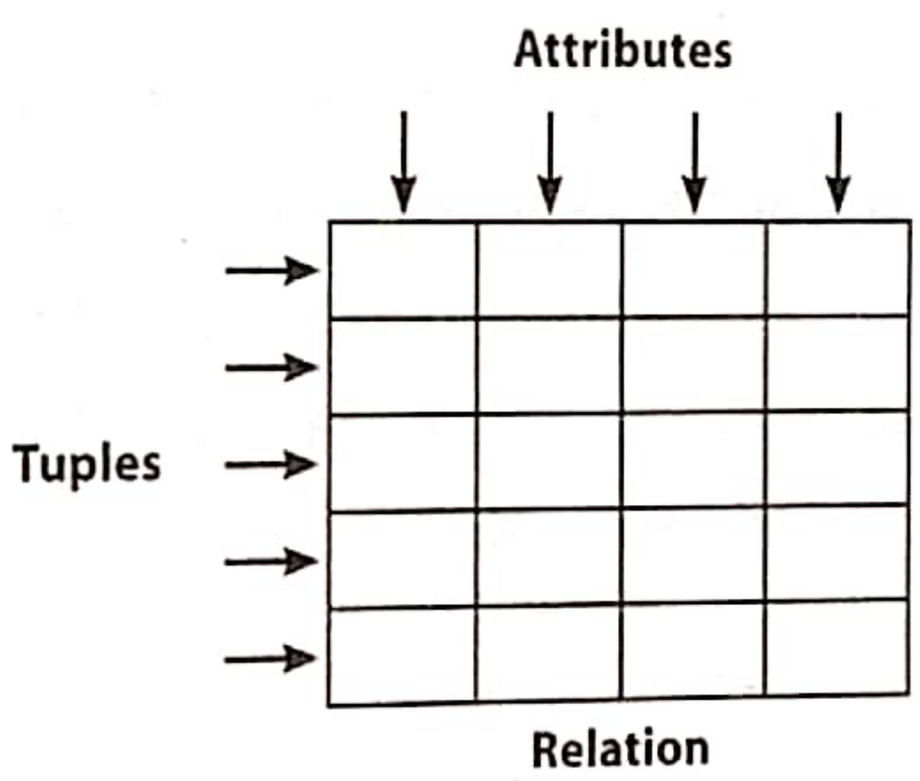


Fig. 10.8: Tuples and Attributes

Basic Terminologies related to a Relational Database

1. **Entity:** An entity is something that exists and about which we can store some information. It is an object which can be distinctly identified. *For example*, student entity, employee entity, item entity, etc. Entity becomes the name of the table.
2. **Attributes:** In a Relational table, an attribute is a set of values of a particular type. The term attribute is also used to represent a column. A table consists of several records (rows), and each record can be broken into several smaller entities known as fields or attributes or columns. A set of attributes defines the characteristics or properties of an entity. In the given table, Student relation consists of four fields or attributes—Roll number, Name, Address and Gender.
3. **Tuple:** Each row in a table is known as tuple. It is also called a row/record. A single entry in a table is called a Record or Row. A Record in a table represents a set of related data. *For example*, the Student table, shown in Fig. 10.9, has 10 records.
4. **Cardinality of Relation:** It is the number of records or tuples in a relation. Thus, the cardinality of Student relation is 10.
5. **Degree of Relation:** Number of columns or attributes is known as the degree of a relation. Thus, the degree of Student relation is 4.

6. **Domain of Relation:** It defines the kind of data represented by the attribute. It is the set of all possible values that an attribute may contain. *For example*, in the given table Student, domain for the field Gender is two since it can have either 'M' or 'F' as the possible and available values that it may contain.
7. **Body of the Relation:** It consists of an unordered set of 0 or more tuples.

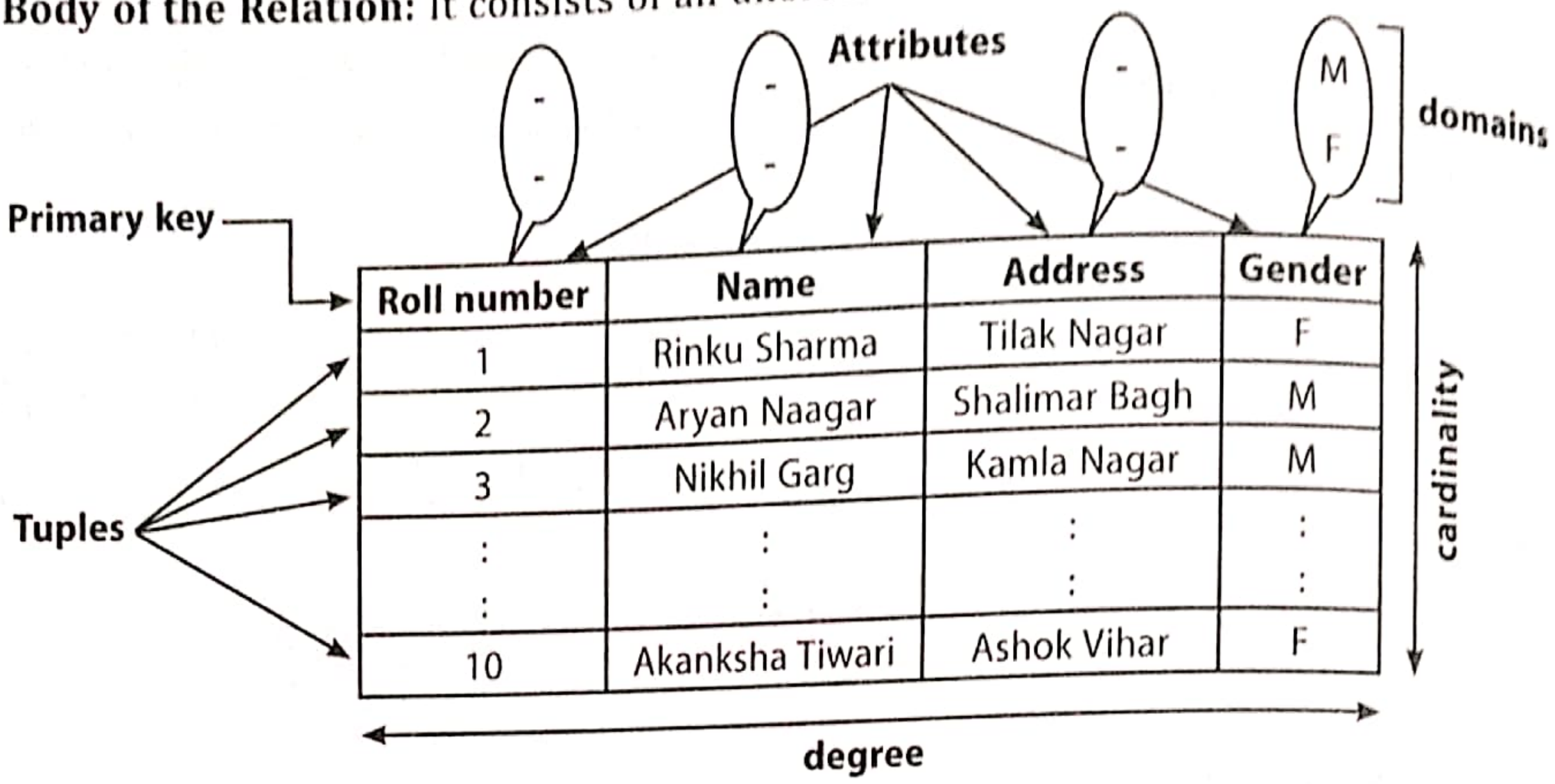


Fig. 10.9: Relational Model (Student-relation)

Therefore, with reference to Fig. 10.9 in the above Student relation,

- There are 10 tuples (*i.e.*, cardinality=10) and 4 attributes (*i.e.*, degree=4).
- Roll number, Name, Address and Gender are the attribute names.
- The first tuple contains the values (1, "Rinku Sharma", "Tilak Nagar", "F").
- The domain of the attribute Gender is (M,F).

10.5 DATABASE KEYS

Keys are a very important part of Relational database. A key allows us to identify an attribute or a set of attributes on the basis of which a table is identified. They are used to establish and identify relation between two or more tables. They also ensure that each record within a table can be uniquely identified by a combination of one or more fields within a table.

The different types of keys in an RDBMS are as follows:

KEY	DESCRIPTION
Primary Key	A primary key is an attribute or a group of attributes that can uniquely identify tuples within the relation.
Candidate Key	A candidate key is one that is capable of becoming the primary key (*i.e.*, candidate for primary key position).
Alternate Key	A candidate key that is not the primary key is called an alternate key.
Foreign Key	A non-key attribute whose value(s) are derived from the primary key of some other table is known as foreign key in its current table.

Let us discuss these keys in detail.

1. **Primary Key:** A primary key is a set of one or more attributes/fields which uniquely identifies a tuple/row in a table. The salient features of a primary key are as follows:
   (a) It is always unique in nature, *i.e.*, non-redundant. It does not have duplicate values in a relation.

(b) It arranges the table in its own order.

(c) It cannot be re-declared or left null.

(d) One table can have only one primary key; however, primary key can be a combination of more than one field. *For example*, roll number along with admission_no can be combined together and declared as a primary key in the relation Student. In the context of the table Item given below, Item_id is the primary key and Supp_id (supplier id) is the primary key in the table Supplier.

Table: Item

Item_id	Item_name	Qty
101	Printer	400
102	CD	200
104	DVD	150
105	Mouse	300
103	Keyboard	180
109	Cable	500

Primary Key — Candidate Keys — Alternate Key

Table: Supplier

Supp_id	Area
S01	Ashok Vihar
S02	Noida
S04	CP
S05	Punjabi Bagh

Primary Key — Candidate Keys — Alternate Key

Fig. 10.10: Keys in a Database (Tables)

2. **Candidate Key:** All the attributes in a relation that are candidates or are capable of becoming a primary key are the candidate keys.

   In the above Item table, Item_id and Item_name are the candidate keys. Out of these keys, Item_id is the primary key and Item_name becomes the alternate key. Similarly, in the case of Supplier relation, Supp_id and Area are the candidate keys; Supp_id is the primary key and Area becomes the alternate key. Thus, the equation becomes:

   Candidate Keys – Primary Key = Alternate Key

3. **Alternate Key:** A candidate key that is not the primary key is called an alternate key. In other words, any attribute that is a candidate for the primary key, *i.e.*, which is capable of becoming a primary key, but is not a primary key, is an alternate key. *For example*, in a customer table, cust_name is the alternate key. Similarly, in the above table Item, Item_name becomes the alternate key.

4. **Foreign Key:** A foreign key is a non-key attribute whose value is derived from the primary key of another table. In other words, primary key in some other table having a relationship with the current or original table is known as foreign key.

Table: Employee

Emp_id	Emp_name	Desig_code (Foreign Key)
E01	Ankur Mehta	Mgr
E02	Deepika Gupta	Dir
E04	Arnav Bansal	Asst_mgr
E03	Harshit Singh	Acc
E05	Kirti Dubey	Mgr

Table: Department

Desig_code	Designation
Mgr	Manager
Dir	Director
Acc	Accountant
Asst_mgr	Assistant Manager

In the above table, Desig_code is the primary key in the table Department which, when related with the table Employee, becomes a foreign key to it.

Till now, we have discussed all about relational databases. These are maintained and operated upon using SQL as the interface. This chapter deals exclusively with relational databases, its tables and retrieving the data using Structured Query Language (SQL).

10.6 OVERVIEW OF SQL AND MySQL

SQL (Structured Query Language) is a standard language for accessing and manipulating databases. SQL commands are used to create, transform and retrieve information from Relational Database Management Systems and also to create an interface between the user and database. By using SQL commands, one can search any data in the database and perform other functions like create tables, add records, modify data, remove rows, drop table, etc.

Fig. 10.11: Structured Query Language (SQL)

MySQL is an open source and freely available Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). It provides excellent features for creating, storing, maintaining and accessing data stored in the form of databases and their respective tables. A single MySQL database can store several tables at a time and can store thousands of records in it.

Being an open-source software, it can be downloaded freely and easily from *www.mysql.org*. MySQL was developed by a Sweden-based company, MySQL AB, which was later acquired by Sun Microsystems. Sun, in turn, was acquired by Oracle Corporation in 2010. It is fully secured, reliable and fast, and possesses better functionalities than many other commercial RDBMs available in the market.

10.7 FEATURES OF SQL

SQL is the most common language used to create, operate, update, manipulate and communicate with a database.

In 1970, SQL was developed by Donald D. Chamberlin and Raymond F. Boyce at IBM. Thus, SQL is a fourth generation non-procedural language that is used to create, manipulate and process the databases (relations). In other words, these languages describe what data is to be retrieved, inserted, updated or deleted from a database.

It has the following salient features and strong processing capabilities:

- It can retrieve data from a database.
- It can insert records in a database.
- It can update records in a database.
- It can create new databases.
- It can create new tables in a database.
- It can create views in a database.

CTM: SQL (Structured Query Language) is a unified, non-procedural language used for creating, accessing, handling and managing data in relational databases.

10.8 ADVANTAGES OF SQL

SQL has the following advantages:

1. **Ease of use:** It is very easy to learn and use and does not require high-end professional training.
2. Large volume of databases can be handled quite easily.
3. **No coding required:** It is non-procedural and a unified language, *i.e.*, we need not specify the procedures to accomplish a task but only need to give a command to perform the activity.
4. SQL can be linked to most of the other high-level languages which makes it the first choice for database programmers.
5. **Portable:** It is compatible with other database programs like Dbase IV, FoxPro, MS-Access, DB2, MS-SQL Server, Oracle, Sybase, etc.
6. SQL is not a case-sensitive language, *i.e.*, both capital and small letters are recognized and treated equally by it.
7. **Powerful language:** All SQL operations are performed at a prescribed and fixed level, *i.e.*, one SELECT command can retrieve data from multiple rows and one MODIFY command can edit multiple rows at a time. These features make SQL a very powerful language as compared to other languages where one command can process only a single record at a time.
8. **Reliable:** SQL provides a high level of well-defined set of commands that provides the desirable results without any ambiguity.
9. **Freedom of data abstraction:** SQL provides a greater degree of abstraction freedom as compared to any other procedural language.
10. **Complete language for a database:** Apart from being a strong query-processing language, it can also be used to create, insert, delete and control access to data in databases.

10.9 CLASSIFICATION OF SQL STATEMENTS

SQL is a language that is used to interact with the database. The SQL statements or commands that we type are the statements that are regarded as instructions to the database.

SQL provides different types of statements or commands for different purposes. These statements are classified into the following categories:

Fig. 10.12: Classification of SQL Statements

We shall now discuss DDL and DML commands in detail.

10.9.1 Data Definition Language (DDL) Commands

The DDL part of SQL permits database tables to be created or deleted. It also defines indices (keys), specifies links between tables and imposes constraints on tables. It contains the necessary statements for creating, manipulating, altering and deleting the table.

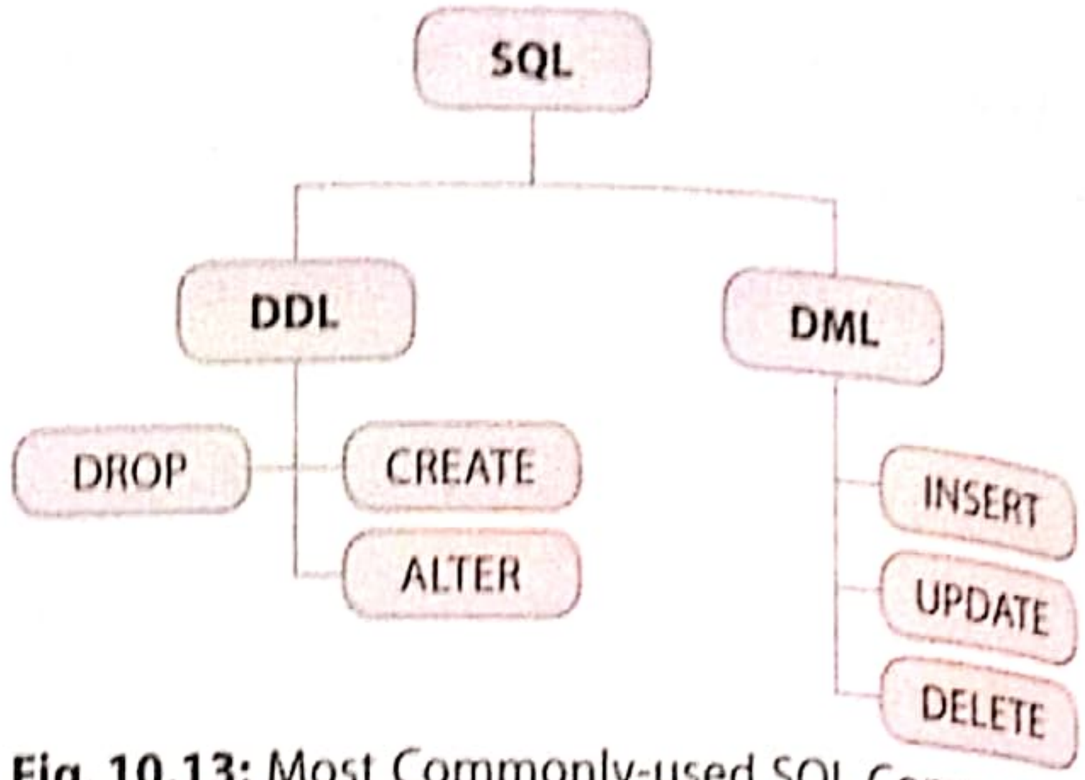


Fig. 10.13: Most Commonly-used SQL Commands

Examples of DDL commands in SQL are:

- **CREATE DATABASE** – creates a new database.
- **USE command** – to select and open an already existing database.
- **CREATE TABLE** – creates a new table.
- **ALTER TABLE** – modifies a table.
- **DROP TABLE** – deletes a table.

CTM: The DDL command lets you define the database structure and its related operations.

DDL provides a set of definitions to specify the storage structure and access methods used by the database system and also defines proper and relevant data types.

10.9.2 Data Manipulation Language (DML) Commands

A Data Manipulation Language (DML) is a part of SQL that helps a user to access or manipulate data. The DML statements are executed in the form of queries which are handled by the DML compiler. It contains a set of statements to:

1. retrieve data from the tables of the database,
2. insert data into the tables of the database,
3. delete data from the tables of the database,
4. update data among the rows/records in the tables of the database.

DML commands carry out query processing operations and manipulate data in the database objects. Several DML commands available are:

1. **SELECT statement**—To extract information from the table. It may or may not be on the basis of certain conditions/criteria.
2. **INSERT INTO statement**—To insert new data (record) into a table.
3. **UPDATE statement**—To modify or change the data (tuple) in a table (not modifying the data type of column).
4. **DELETE**—To delete data (tuple) from a table (not deleting a column).

Learning Tip: The query and update commands form the DML part of SQL. It enables the user to access or manipulate the data stored in a database.

Note: DCL and TCL commands are beyond the scope of this book.

Table 10.1: Difference between DDL and DML Commands

DDL Commands	DML Commands
1. DDL stands for Data Definition Language.	1. DML stands for Data Manipulation Language.
2. These commands allow us to perform tasks related to data definition, *i.e.*, related to the structure of the database objects (relations/databases).	2. These commands are used to manipulate data, *i.e.*, records or rows in a table or relation.
3. Examples of DDL commands are Create, Alter, Drop, Grant, Revoke, etc.	3. Examples of DML commands are Insert into, Update, Delete, Select, etc.
4. DDL is not further classified.	4. DML is further classified into two types: (a) Procedural DMLs (b) Non-procedural DMLs

Let us start implementing SQL using MySQL as the platform.

10.10 MySQL

MySQL is the most popular open source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications.

MySQL database system works on Client/Server architecture. It constitutes a MySQL server which runs on a machine containing the databases and MySQL databases (clients) which are connected to these server machines over a network.

Advantages of MySQL

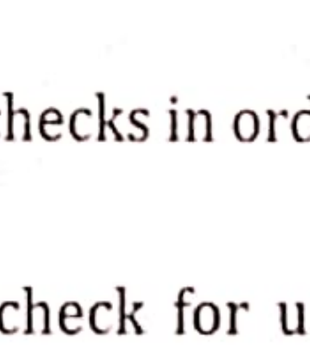
MySQL has the following salient features and advantages:

1. **Reliability and Performance:** MySQL is a very reliable and high performance Relational Database Management System.
2. **Modifiable:** Being an open source software, MySQL comes with its source code, so it is easily modifiable and we can recompile its associated source code.
3. **Multi-platform Support:** MySQL supports several different platforms like UNIX, Linux, Mac OS X and Microsoft Windows.
4. **Powerful Processing Capabilities:** MySQL is a powerful, easy, compatible and fast Relational Database Management System. It can handle complicated corporate applications and processing requirements.
5. **Integrity (Checks):** MySQL provides various integrity checks in order to restrict user input and processing.
6. **Authorization:** MySQL provides DDL commands to check for user authentication and authorization by restricting access to relations and views.

10.10.1 Starting MySQL Database

MySQL is an open source database system. You can download and install it directly from the internet. After installing, you need to start working with MySQL by following the given steps:

Start
→ All Programs
→ MySQL
→ MySQL Server
→ MySQL Command Line Client



Alternatively, (for Windows 7 and above):

Start -> Apps by name -> MySQL Command Line Client (Fig.10.14)

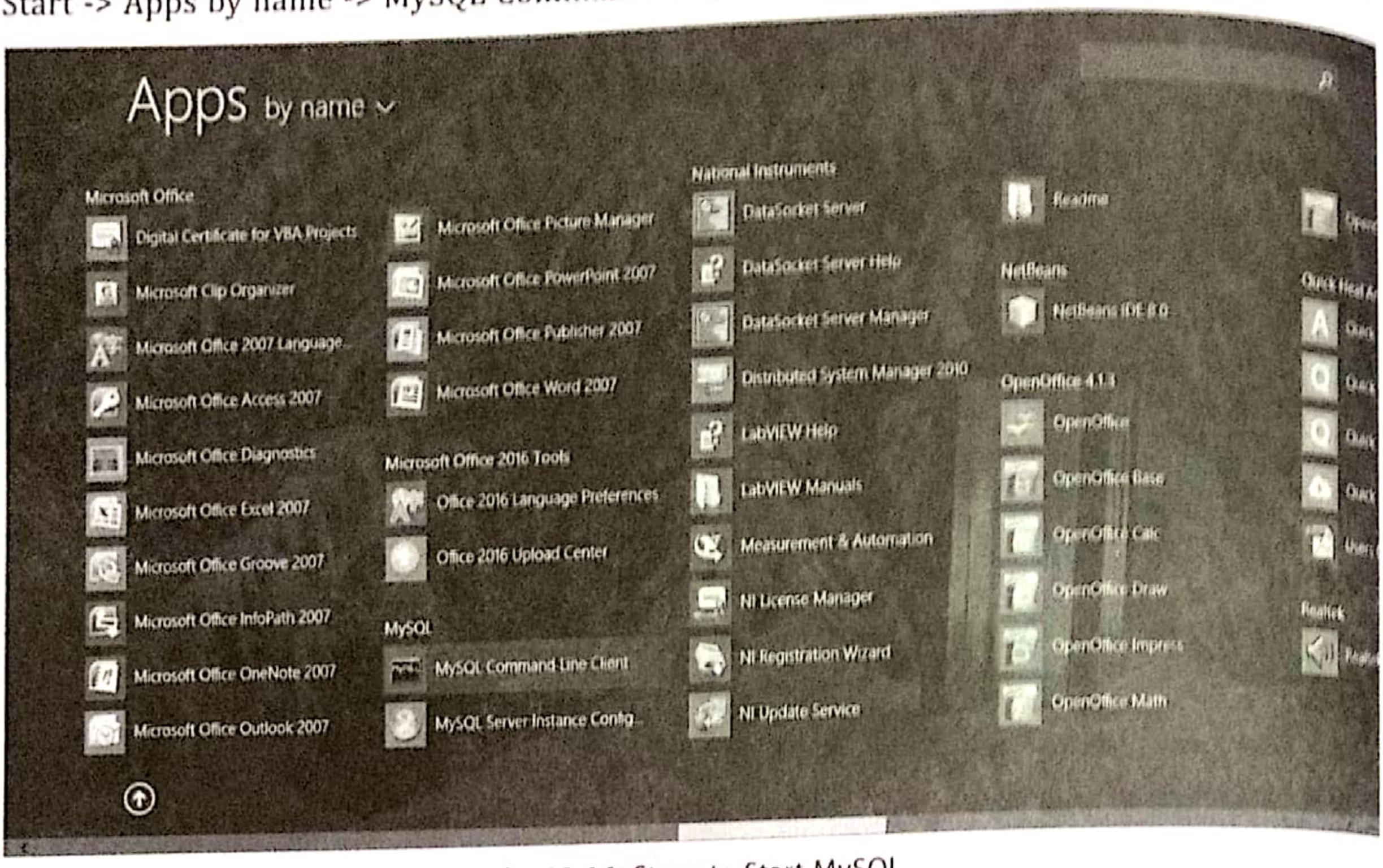


Fig. 10.14: Steps to Start MySQL

After opening MySQL, the screen of the MySQL command prompt appears where you need to specify a password to work with it.

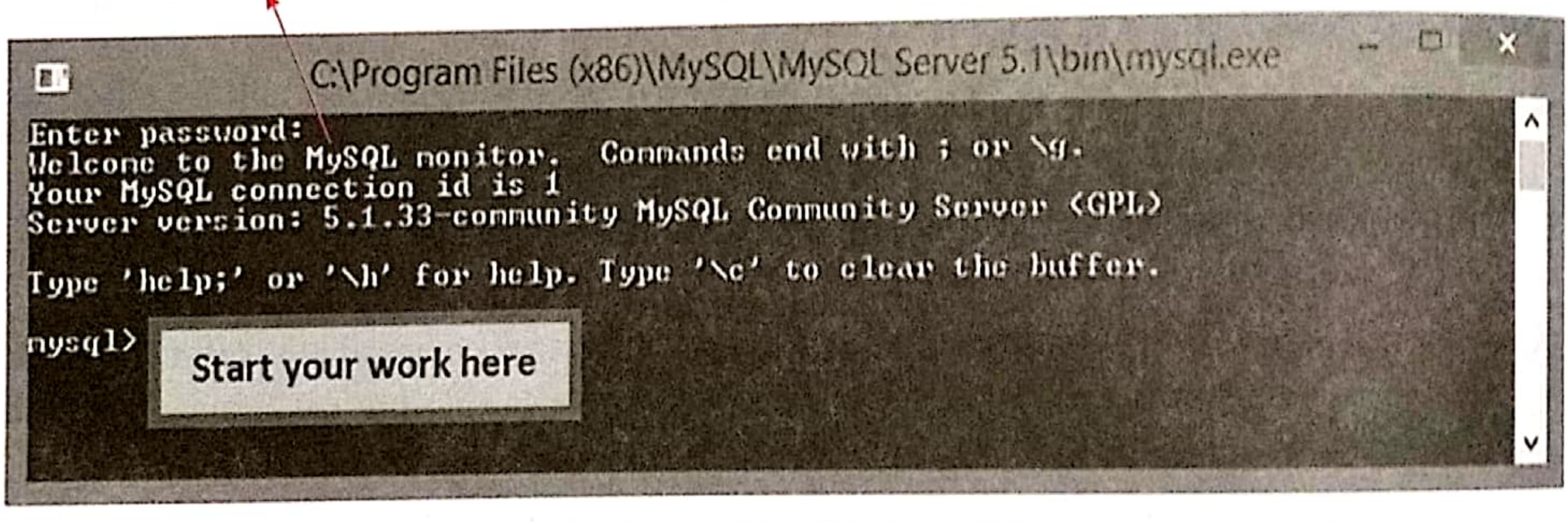


Fig. 10.15: MySQL Prompt Screen

After entering the password, the MySQL prompt appears where you start typing the SQL commands as shown in Fig. 10.15.

In order to come out of MySQL application, you can type *quit* in front of the *mysql>* command prompt as shown in Fig. 10.16.

Learning Tips:

- Some database systems require a semicolon (;) at the end of each SQL statement.
- Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.
- SQL is NOT case-sensitive; select is the same as SELECT.

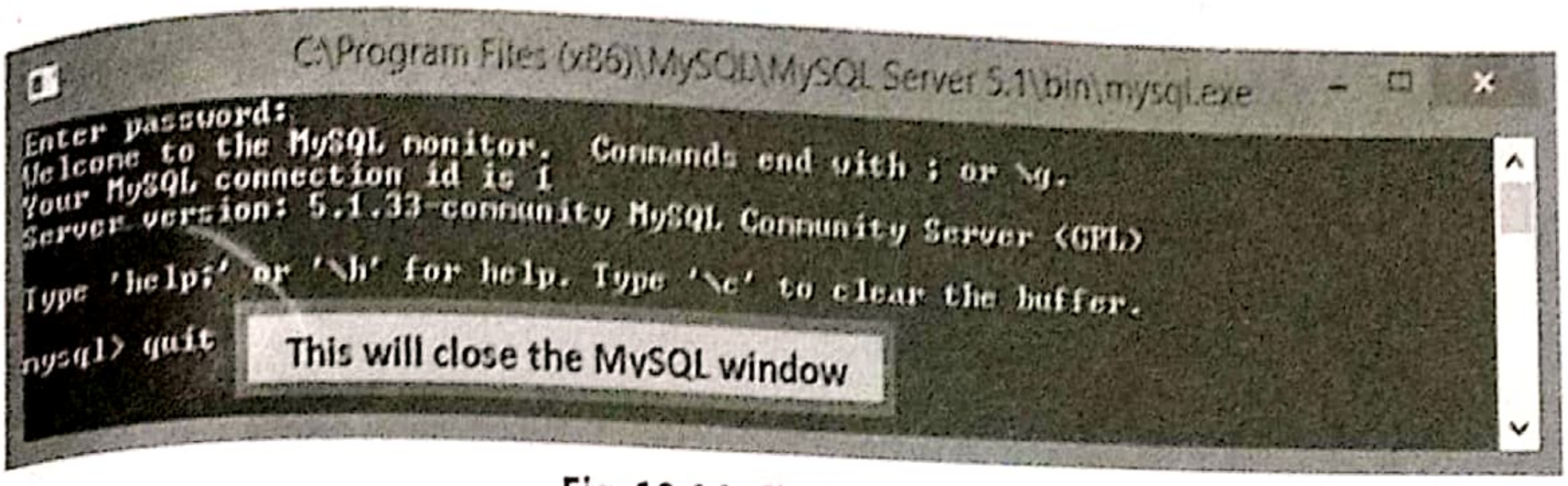


Fig. 10.16: Closing MySQL

10.11 SQL DATATYPES

Just like any other programming language, the facility of defining data of various types is available in SQL also. SQL supports the following data types for the specification of various data-items or fields of a relation/table. In SQL, each column of the table is assigned a data type which conveys the kind of value that will be stored in the column.

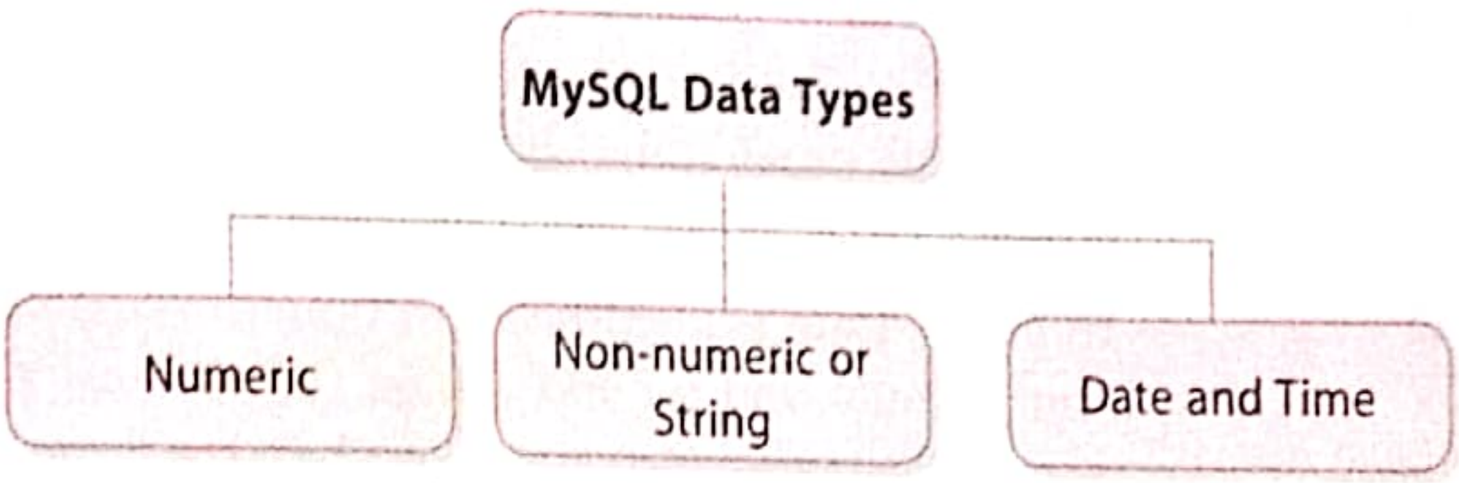
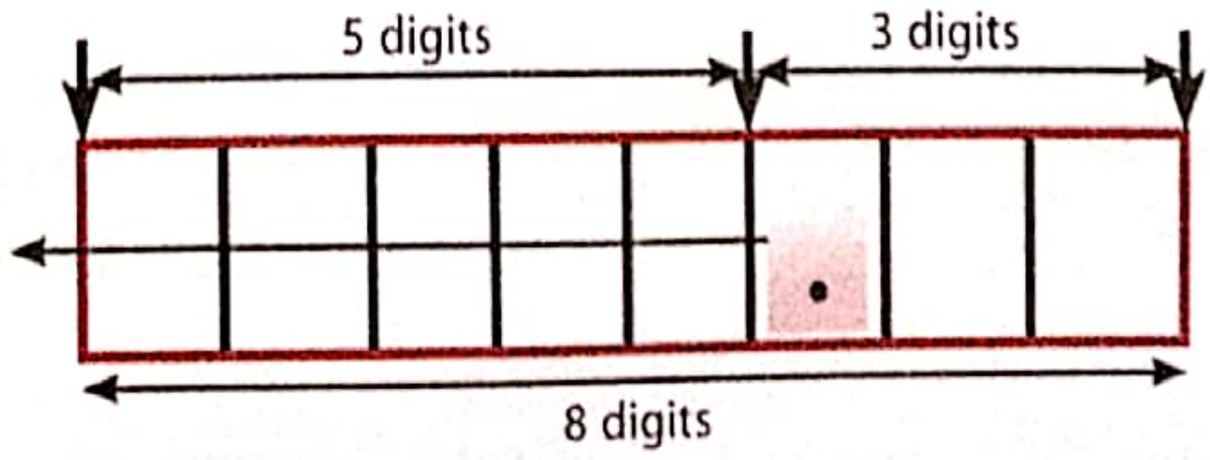


Fig. 10.17: MySQL Data Types

Data Type	Syntax	Description
INTEGER (Numeric)	**INTEGER or integer**	It stores/represents positive whole numbers up to 11 digits and negative whole numbers up to 10 digits. The range of integer is from –2,147,483,648 to 2,147,483,647.
SMALLINT	**SMALLINT**	It is a 16-bit signed integer value that stores whole numbers in the range from -32768 to 32767. Its width is up to 5 digits.
NUMERIC	**NUMERIC(x,y)**	Numbers are stored in the given format, where x is the total number of digits and y is the number of places to the right of the decimal point. x must include an extra place for the decimal point. *For example*, Numeric(8,2). In the given example, numeric data type stores a number that has 5 places before the decimal and 2 digits after the decimal and 1 digit place for the decimal point. Numeric holds up to 20 significant digits. A negative number holds one place for the sign, *i.e.*, (–).
DECIMAL	**DECIMAL(x,y) Or DECIMAL(size, precision)**	Numbers stored in the DECIMAL format, where x is the size, *i.e.*, total number of digits and y is precision, *i.e.*, it is the number of places to the right of the decimal point. *For example*, Decimal(8,2). In the above example, decimal data type stores a number that has 5 digits before the decimal and 2 digits after the decimal and 1 digit place for the decimal. Decimal holds up to 19 significant digits. A negative number uses one place for its sign (–).

decimal point



CHARACTER (fixed length)	**CHAR(x) or CHAR(size)**	This data type stores 'x' number of characters in the string which has a fixed length. A maximum of 254 characters can be stored in a string. If you store strings that are not as long as the 'size' or 'x' parameter value, the remaining spaces are left unused. *For example*, if you specify CHAR(10), strings such as "ram" and "technology" are each stored as 10 characters. However, a student admission_no is 6 digits long in a school, so CHAR(6) would be appropriate to store the admission_no of all the students. This data type is suitable where the number of characters to be stored is fixed. The value for CHAR data type has to be enclosed in single or double quotation marks.
CHARACTER (variable length)	**VARCHAR(x) or VARCHAR2(x)**	This data type is used to store variable length alphanumeric data. *For example*, address of a student can be declared as VARCHAR(25) to store the address up to 25 characters long. The advantage of using this data type is that VARCHAR will not leave unused spaces. It releases the unused memory spaces.
DATE	**DATE**	This data type is used to store a date in 'yyyy/mm/dd' format. It stores year, month and date values. DATE values can be compared with each other only. The date values to be entered are to be enclosed in { } or with single quotation marks.
TIME	**TIME**	This data type is used to store time in hh:mm:ss format. It stores hour, minute and second values. *For example*, time of the day can be taken as 12:30:45p.m. where 12 means hours, 30 means minutes and 45 are seconds.
BOOLEAN (logical)	**BOOLEAN**	This data type is used for storing logical values, either true or false. In both upper and lower case, T or Y stands for logical true and F or N stands for logical false. The fields with Boolean (logical) data type can be compared only with to other logical columns or constants.
BLOB/RAW/ LONG RAW	**BLOB or RAW or LONG RAW**	This data type can store data up to a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and used to store a large amount of data, such as images, animation, clips or other types of files. *For example*, image raw(2000);
MEMO/LONG	**MEMO or LONG**	This data type allows storing characters or remarks up to 2 GB per record.

Table 10.2: Difference between CHAR and VARCHAR Data Types.

CHAR	VARCHAR
1. CHAR data type provides fixed-length memory storage. It specifies a fixed-length character string.	1. VARCHAR data type provides variable-length memory storage. It specifies a variable-length string (changeable).
2. The CHAR data type can store a maximum of 0 to 255 characters.	2. The VARCHAR data type can store a maximum of 65,535 characters.
3. CHAR data type is used when the data entries in a column are expected to be of the same size.	3. VARCHAR data type is used when the data entries in a column are expected to vary considerably in size.
4. CHAR(x) will take x characters of storage even if you enter less than x characters in that column.	4. VARCHAR(x) will take only the required storage for the actual number of characters entered in that column.

5. If a value entered is shorter than its length x, then blanks are added.
6. CHAR data type takes memory space of 1 byte per character.
7. Search operation is faster with CHAR data type column.
8. *For example*, name char(10);
   name="anu";
   name field occupies 10 bytes, with the first three bytes for values and the rest for blank data.

5. No blanks are added if the length is shorter than the maximum length x.
6. VARCHAR takes up memory space of 1 byte per character, +2 bytes to hold variable-length information.
7. Search operation works slower in VARCHAR data type column as compared to CHAR type.
8. *For example*, name varchar(10);
   name="anu";
   then name occupies only 3+2=5 bytes, first three bytes for value and the other two bytes for variable length information.

CTM: While defining data type for columns or attributes in a relation, two points should be kept in mind:
1. When using fixed-length data in columns, like phone number, area code, use character data type.
2. When using variable-length data in columns, like name, address, designation, use varchar data type.

10.12 SQL COMMANDS

Let us now learn how a database and tables in the database are created in SQL. A database is used to house data in the form of tables. Therefore, before creating a table, it is mandatory to create a database first.

We shall create a database, *school*, in which there are two tables, namely *student* and *fees*.

Database: School

Tables_in_School
Student
Fees

Table: Student

Rollno	Name	Gender	Marks	DOB
1.	Raj Kumar	M	93	17-Nov-2000
2.	Deep Singh	M	98	22-Aug-1996
3.	Ankit Sharma	M	76	02-Feb-2000
4.	Radhika Gupta	F	78	03-Dec-1999
5.	Payal Goel	F	82	21-April-1998
6.	Diksha Sharma	F	80	17-Dec-1999
7.	Gurpreet Kaur	F	65	04-Jan-2000
8.	Akshay Dureja	M	90	05-May-1997
9.	Shreya Anand	F	70	08-Oct-1999
10.	Prateek Mittal	M	75	25-Dec-2000

Table: Fees

Rollno	Name	Fees
1.	Raj Kumar	8000
2.	Deep Singh	9000
5.	Payal Goel	7500
6.	Diksha Sharma	8000
9.	Shreya Anand	8020
10.	Prateek Mittal	9200

To get started with our own database, we can first check which databases currently exist in MySQL server. Use the ***SHOW DATABASES*** statement to find out which databases currently exist on the server:

```
mysql> show databases;
+--------------+
| Database |
+--------------+
| mysql|
| test |
+--------------+
2 rows in set (0.01 sec)
```

1. Creating Databases

The ***CREATE DATABASE*** command is used to create a database in RDBMS.

Syntax for creating a database:

CREATE DATABASE <database_name>;

For example,

CREATE DATABASE school; ⟵ Creates database with the name school.

When the above-mentioned command gets executed, a database with the name school will be created on the system.

2. Opening Databases

Once a database has been created, we need to open it to work on it. For this, ***USE*** command is required.

Syntax for opening a database:

USE <database_name>;

For example,

***mysql*>USE school;**

Database changed

3. Removing Databases

To physically remove/delete a database along with all its tables, ***DROP*** command is used.

Syntax for removing a database:

DROP DATABASE <database_name>;

For example,

***mysql*>DROP DATABASE school;**

Database deleted

4. Creating a Table

The **CREATE TABLE** statement is used to create a table in a database. Tables are organized into rows and columns, and each table must have a name. It is the most extensively used DDL command. A table must have at least one column.

Syntax for creating a table:

CREATE TABLE *<table_name>*

(

<column_name1><data_type> [size],

<column_name2><data_type> [size],

<column_name3><data_type> [size],

....

);

For example,

```
mysql>CREATE TABLE student
( Rollno        integer        NOT NULL PRIMARY KEY,
  Name          varchar(20)    NOT NULL,
  Gender        char(1),
  Marks         integer(11),
  DOB           date );
```

Query OK, 0 rows affected (0.04 sec)

For each column, a name and a data type must be specified and the column name must be unique within the table definition. Column definitions are separated by a comma. Upper case and lower case letters make no difference in column names, the only place where upper and lower case letters matter are string comparisons.

5. Viewing a Table

To verify that the table has been created, ***SHOW TABLES*** command is used.

```
mysql> show tables;
+----------------------+
| Tables_in_school  |
+----------------------+
| student              |
| fees                 |
+----------------------+
2 rows in set (0.01 sec)
```

6. Viewing a Table Structure

To view a table structure, ***DESCRIBE or DESC*** command is used.

Syntax: ***DESCRIBE <tablename>; or DESC <tablename>;***

For example,

mysql> describe student;

Field	Type	Null	Key	Default	Extra
Rollno	integer(11)	YES		NULL	
Name	varchar(20)	YES		NULL	
Gender	char(1)	YES		NULL	
Marks	number(11)	YES		NULL	
DOB	date	YES		NULL	

5 rows in set (0.02 sec)

7. Inserting Data into a Table

The ***INSERT INTO*** command is used to insert a new record/row/tuple in a table.

It is possible to write the INSERT INTO statement in the following formats:

(a) **Inserting the data (for all the columns) into a table:** In the first method, it does not specify the column names where the data will be inserted, only their values.

Syntax for SQL INSERT is:

Syntax: INSERT INTO *table_name*

VALUES (*value1, value2, value3...***);**

For example, INSERT INTO student

VALUES (1,"Raj Kumar", 'M', 93, '2000-11-17');

While inserting a row, if you are adding value for all the columns of the table, you need not specify the column(s) name in the SQL query. But you need to make sure that the order of the values is in the same order as the columns represented in the structure of the table. The following points should be kept in mind while inserting data in a relation:

- When values are inputted using INSERT INTO command, it is termed as single row insert since it adds one tuple at a time into the table.
- The INTO clause specifies the target table and the VALUES clause specifies the data to be added to the new record of the table.
- The argument/values of character data type are always enclosed in double or single quotation marks.
- Column values for the data type of a column are provided within curly braces { } or single quotes.
- NULL values are stored and displayed as NULL only without any quotes.
- If the data is not available for all the columns, then the column-list must be included following the table name.

CTM: In SQL, we can repeat or re-execute the last command typed at SQL prompt by typing "/" key and pressing enter.

(b) **Inserting the data directly into a table:** The second form specifies both the column names and the values to be inserted.

Syntax: **INSERT INTO** table_name **(**column1,column2,...columnN,**)**
VALUES (value1,value2,value3,...valuen**);**

Here, column1, column2,...columnN—the names of the columns in the table for which you want to insert data.

For example, **INSERT INTO student(Rollno, Name, Gender, Marks, DOB)**
VALUES(2,'Deep Singh', 'M', 98, '1996-08-22');

CTM: When adding a row, only the characters or date values should be enclosed within single quotes.

(c) **Inserting data into a table through a select statement (using a temporary table):**

Syntax for SQL INSERT is:

INSERT INTO <table_name>[(column1, column2, ... columnN)]

SELECT <column1>,< column2>, ...<columnN>

FROM <table_name> [WHERE condition];

For example, to insert a row into the student table from a temporary table, the SQL insert query is:

INSERT INTO student (Rollno, Name, Gender, Marks, DOB) SELECT stud_rollno, stud_name, stud_marks FROM temp_student where Marks>80;

The above statement shall extract all those records from temp_student whose marks are more than 80 and shall place these matching records into the temporary table. To insert using the SQL query, the following two conditions must be true:

1. Both the tables must already be created.
2. The columns being inserted must match the columns output by the subquery.

If you are inserting data into all the columns, the above insert statement can also be written as:

INSERT INTO student

SELECT * FROM temp_student;

POINT TO REMEMBER

We have assumed the temp_student table has columns stud_rollno, stud_name, gender, stud_marks, DOB in the above given order and with the same data type.

(d) **Inserting NULL values into a table:** NULL values are treated differently from other values as they represent missing unknown data. By default, a column in a table can hold NULL values. If a column in a table is optional, we can insert a new record or can modify an existing tuple without adding values to this column. In other words, the values in every record for this column/field shall be stored as NULL. We can insert NULL value into any column in a table. It can be done by typing NULL without quotes.

Null is not equivalent to 0, *i.e.*, NULL ≠ 0. It acts as a placeholder for unknown or inapplicable values.

For example, **INSERT INTO student(Rollno, Name, Gender, Marks, DOB)**
VALUES(12, 'Swati Mehra', 'F', NULL, NULL);

After the execution of the above command, NULL value shall be inserted for the fields Marks and DOB respectively.

(e) **Inserting data interactively into a table:** For inserting a row interactively (from the keyboard), & operator can be used.

For example, INSERT INTO student
VALUES('&Roll_no', '&Name', '&Gender', '&Marks', '&DOB');

In the above command, the values for all the columns are read from the keyboard and inserted into the table student.

8. Modifying data in a Table

To modify data in a table or to make changes for some or all of the values in the existing records in a table, we use the *UPDATE* statement. The UPDATE command specifies the rows to be modified using the WHERE clause and the new data is written into the respective record using the SET keyword.

Syntax for UPDATE:

UPDATE <table_name>
SET <column1> = <value1>, <column2> = <value2>,.....
WHERE <column_name> = <new_value>;

For example, UPDATE student
SET Marks = 90
WHERE Rollno = 8;

The above statement shall change the value of Marks field to 90 for the student whose roll number is 8.

(a) **Updating multiple columns**

Modifying the values in more than one column can be done by separating the columns along with the new values using SET clause, separated by commas.

For example, UPDATE student
SET Marks = 70, DOB='1998-08-11'
WHERE Name="Payal";

The above statement shall change the values for both the fields—Marks and DOB—to 70 and '1998-08-11' respectively for the student whose name is Payal.

(b) **Updating to NULL values**

The values for the attributes in a relation can also be entered as NULL using UPDATE command.

For example, UPDATE student
SET Marks = NULL
WHERE Rollno = 9;

The above statement shall change the value of the field Marks to 0 for the student whose roll number is 9.

(c) **Updating using an expression or formula:**

An expression can also be used with UPDATE statement to change the values for an attribute.

For example,

```
UPDATE student
SET Marks = Marks + 10
WHERE (Rollno = 5 or Rollno = 10);
```

The above statement shall increment the value of Marks by 10 for all the records with Roll number 5 or 10.

9. Removing data from a table

The ***DELETE*** statement is used to delete rows from a table.

Syntax for DELETE Statement:

DELETE FROM <table_name> [WHERE <condition>];

where <table_name> is the table whose records are to be deleted.

POINT TO REMEMBER

The *WHERE* clause in the SQL delete command is optional and identifies the rows in the column that get deleted. If you do not include the WHERE clause, all the rows in the table are deleted.

For example,

DELETE FROM student WHERE Rollno = 10;

The above statement shall delete the record only for roll number 10.

To delete all the rows from the student table, the DELETE statement will be:

DELETE FROM student;

➢ SQL TRUNCATE Statement

The ***SQL TRUNCATE*** command is used to delete all the rows from the table and free the space containing the table.

Syntax to TRUNCATE a table:

TRUNCATE TABLE<table_name>;

For example,

To delete all the rows from student table, the statement will be:

TRUNCATE TABLE student;

Difference between DELETE and TRUNCATE Statements

DELETE Statement: This command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.

TRUNCATE Statement: This command is used to delete all the rows from the table and free the space containing the table.

10. ALTER TABLE Command

The ***ALTER TABLE*** command is used to modify the definition (structure) of a table by modifying the definition of its columns. The ALTER TABLE command is used to perform the following operations:

- To add a column to an existing table.
- To rename any existing column.
- To change the data type of any column or to modify its size.
- To remove or physically delete a column.

A. Adding a column to an existing table

Once a table has been created, new columns can be added later on, if required. The new column is added with NULL values for all the records/rows in the table. It is possible to add, delete and modify columns with ALTER TABLE statement.

Syntax for adding a new column:

ALTER TABLE <table_name> ADD(<column_name><datatype> [size]);

For example, to add a new column Mobile_no of type integer in the table student:

ALTER TABLE student ADD (Mobile_no integer);

Thus, the above statement shall add a new column Mobile_no to the table student with NULL value in it.

POINT TO REMEMBER

We have just added a column and there will be no data (NULL) under this attribute. UPDATE command can be used to supply values/data to this column.

B. Renaming a column (using MODIFY clause)

To rename a column of a table, Modify column clause with ALTER TABLE can be used.

Syntax for renaming a column:

ALTER TABLE <table_name>

MODIFY [column_name] <old_col_name><new_col_name> [column definition];

For example,

ALTER TABLE student

MODIFY DOB date_of_birth date;

The above command shall change the name of DOB field to date_of_birth as the new name.

Resultant table: student

Rollno	Name	Gender	Marks	Date_of_Birth	Mobile_no
1.	Raj Kumar	M	93	17-Nov-2000	NULL
2.	Deep Singh	M	98	22-Aug-1996	NULL
3.	Ankit Sharma	M	76	02-Feb-2000	NULL

C. Adding a column with Default Value

ALTER TABLE command can be used to add a new column to an existing table with default values.

Syntax for adding a column with a default value:

ALTER TABLE <table_name>

ADD ([column_name1]<data type1>default data);

For example, ALTER TABLE student ADD(City char(6) DEFAULT "DELHI");

The above command will add a new column City with default value as "DELHI" to the student table.

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	City
1.	Raj Kumar	M	93	17-Nov-2000	NULL	DELHI
2.	Deep Singh	M	98	22-Aug-1996	NULL	DELHI
3.	Ankit Sharma	M	76	02-Feb-2000	NULL	DELHI
4.	Radhika Gupta	F	78	03-Dec-1999	NULL	DELHI
5.	Payal Goel	F	82	21-April-1998	NULL	DELHI
6.	Diksha Sharma	F	80	17-Dec-1999	NULL	DELHI
7.	Gurpreet Kaur	F	65	04-Jan-2000	NULL	DELHI
8.	Akshay Dureja	M	90	05-May-1997	NULL	DELHI
9.	Shreya Anand	F	70	08-Oct-1999	NULL	DELHI
10.	Prateek Mittal	M	75	25-Dec-2000	NULL	DELHI

D. Modifying an existing column definition

The MODIFY clause can be used with ALTER TABLE command to change the data type, size, constraint related to any column of the table.

Syntax for modifying existing column data type:

ALTER TABLE <table_name>

MODIFY([column_name1] <datatype1>);

For example,

ALTER TABLE student MODIFY (Name varchar(25));

The above command will modify the data type size for the Name field from 20 to 25 characters.

E. Renaming a column (using RENAME clause)

The existing column in a relation can be renamed using ALTER TABLE command.

Syntax for renaming an existing column:

ALTER TABLE <table_name>

RENAME <old_column_name> to <new_column_name>;

For example,

ALTER TABLE student

RENAME City to State;

The above command shall rename the City column to State.

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	State
1.	Raj Kumar	M	93	17-Nov-2000	NULL	DELHI
2.	Deep Singh	M	98	22-Aug-1996	NULL	DELHI
3.	Ankit Sharma	M	76	02-Feb-2000	NULL	DELHI
4.	Radhika Gupta	F	78	03-Dec-1999	NULL	DELHI
5.	Payal Goel	F	82	21-April-1998	NULL	DELHI
6.	Diksha Sharma	F	80	17-Dec-1999	NULL	DELHI
7.	Gurpreet Kaur	F	65	04-Jan-2000	NULL	DELHI
8.	Akshay Dureja	M	90	05-May-1997	NULL	DELHI
9.	Shreya Anand	F	70	08-Oct-1999	NULL	DELHI
10.	Prateek Mittal	M	75	25-Dec-2000	NULL	DELHI

F. Removing a column

To remove or drop a column in a table, ALTER TABLE command is used.

Syntax for removing a column:

ALTER TABLE <table_name>

DROP <column_name>;

For example,

ALTER TABLE student DROP (State);

The above command will drop State column from the student table.

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no
1.	Raj Kumar	M	93	17-Nov-2000	NULL
2.	Deep Singh	M	98	22-Aug-1996	NULL
3.	Ankit Sharma	M	76	02-Feb-2000	NULL
4.	Radhika Gupta	F	78	03-Dec-1999	NULL
5.	Payal Goel	F	82	21-April-1998	NULL
6.	Diksha Sharma	F	80	17-Dec-1999	NULL
7.	Gurpreet Kaur	F	65	04-Jan-2000	NULL
8.	Akshay Dureja	M	90	05-May-1997	NULL
9.	Shreya Anand	F	70	08-Oct-1999	NULL
10.	Prateek Mittal	M	75	25-Dec-2000	NULL

11. DROP TABLE Command

Sometimes we may need to physically remove a table which is not in use. ***DROP TABLE*** command is used to remove/delete a table permanently. It should be kept in mind that we cannot drop a table if it contains records. That is why the rows of the table have to be deleted first and only then can the table be dropped. This command will completely destroy the table structure.

Syntax for removing a table:

DROP TABLE <table_name>;

For example,

mysql>DROP TABLE student;

This command will permanently remove the table student from the database school.

10.13 INTEGRITY CONSTRAINTS

You are already familiar with the Data Definition Language (DDL) commands; apart from dealing with the structure of a table, these commands allow you to control its contents with the help of constraints. Constraints are defined as the checks or conditions that are applied to the columns or fields of the table.

Constraints, also termed as restrictions, are imposed on the user, so that the input can be masked according to the database structure. Constraints are also known as Integrity Constraints because they are implemented to maintain data integrity in a database.

CTM: Integrity constraints or Constraints are the set of rules, conditions or checks applicable to a column or table which ensures the integrity or validity of data.

Table 10.1: Constraints in MySQL

S.No.	Constraint	Description/Purpose
1.	PRIMARY KEY	This constraint sets a column or group of columns as the primary key of the table. It identifies a record uniquely and distinguishes one record from another in a table. Therefore, NULL values and Duplicate values are not accepted in the column defined as the primary key.
2.	FOREIGN KEY	This contraint identifies any column referencing (pointing to) the PRIMARY KEY in another table. A foreign key is always a primary key from another table. Therefore, data will be accepted in this column if same data value exists in a column (Primary Key) in another related table. It ensures Referential Integrity of the data.
3.	NOT NULL	This constraint ensures that the specified column does not accept a NULL value or cannot have a NULL value.
4.	UNIQUE	This constraint ensures that duplicate values are not accepted in the specified columns, *i.e.*, all the values in the specified column are different.
5.	CHECK	This constraint ensures that all the values being entered in the specified column must satisfy a certain condition specified with it.
6.	DEFAULT	This constraint provides a default value for a column when no value is given.

We shall now discuss the PRIMARY KEY and FOREIGN KEY constraints in detail. Rest of the constraints are beyond the scope of this book.

10.13.1 PRIMARY KEY

As described above (Table 10.1), Primary key of a table is a column or group of columns that uniquely identifies a record/row/tuple in a table. PRIMARY KEY constraint means that a column cannot have duplicate values and not even a null value. This constraint is used to define a column in a table as its primary key. Primary key helps to distinguish one record (row or tuple) from another in a relation, since it restricts the user from entering any duplicate values in it. Therefore, no two rows of a table can have the same primary key value.

Now, suppose you created a table **student** with the following statement:

mysql>CREATE TABLE student

(Rollno integer, Name varchar (20), Gender char (1),

Marks integer (11));

We know that Rollno in this table is a primary key. But MySQL does not understand this unless we explicitly mention it while creating the table. Since it is not treated as a primary key by MySQL compiler, it is possible to enter duplicate values or NULL values in this column, which will be wrong and shall nullify the significance of having a primary key in a table.

In order to ensure that such data is not accepted by MySQL, we can define Rollno as the primary key at the time of creation of the table using the PRIMARY KEY clause as described below:

mysql>CREATE TABLE student **(Defining Primary Key at Column Level)**

(Rollno integer PRIMARY KEY, Name varchar (20) ,

Gender char(1) , Marks integer(11));

Alternatively,

mysql>CREATE TABLE student **(Defining Primary Key at Table Level)**

(Rollno integer, Name varchar (20),

Gender char(1), Marks integer(11), PRIMARY KEY(Rollno);

Now, suppose we have another field Admn_no (admission number of the student), added later to this table, which can be a better candidate to be the primary key. In such a case, if we give the statement as:

mysql>CREATE TABLE student

(Rollno integer PRIMARY KEY,

Admn_no integer PRIMARY KEY,

Name varchar (20),

Gender char(1), Marks integer(11));

then, it shall generate an error (Multiple primary key defined) because it will be treated as two separate primary keys by MySQL, which is not permitted. A table can have at most one primary key. This problem can be resolved by taking the combination of Admn_no and Rollno as the primary key, given by the statement as follows:

mysql>CREATE TABLE student

(Admn_no integer , Rollno integer,

Name varchar (20),

Gender char(1), Marks integer(11)

PRIMARY KEY(Admn_no, Rollno));

This can be verified by checking the structure of the table with the command as:

mysql> DESC student;

Field	Type	Null	Key	Default	Extra
Admn_no	integer(11)	NO	PRI	0	
Rollno	integer(11)	NO	PRI	0	
Name	varchar(20)	YES		NULL	
Gender	char(1)	YES		NULL	
Marks	number(11)	YES		NULL	
DOB	date	YES		NULL	

This combination of two constitutes the primary key of the table.

NULL values cannot be accepted in these columns.

10.13.2 FOREIGN KEY

This constraint identifies any column referencing the PRIMARY KEY in another table. A foreign key is always a primary key from another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a foreign key, it should be defined as a primary key in the table which it is referring to. One or more columns can be defined as foreign key.

This constraint requires two tables in which Reference table (having Primary Key) is called the Parent table and the table having foreign key is called the Child table. Thus, foreign key is used to enforce referential integrity since it is declared as a primary key in some other table.

Syntax to define a foreign key:

[CONSTRAINT (column_name) constraint_name] REFERENCES

References_Table_name(column_name)

In the above syntax,

- The first column_name is the name of the related column in the child table.
- The column_name appearing after REFERENCES clause is the name of the related column in the parent table.
- The table_name after REFERENCES clause is the name of the parent table to which the column in current table is related.

For example,

```
mysql>CREATE TABLE student <------ Parent table
    (Rollno      integer        NOT NULL PRIMARY KEY,
    Name         varchar(20)    NOT NULL,
    Gender       char(1),
    Marks        integer(11));
```

```
mysql>CREATE TABLE fees          <-- Child table
    (Rollno      integer         FOREIGN KEY(Rollno)
    Name         varchar(20)     NOT NULL,
    Total_fees   number(4),
    Fee_date     date);
```

FOREIGN KEY

Primary key of Parent table

REFERENCES student(Rollno)

Parent Table Name

In the above example, Rollno is the primary key in student table and becomes a foreign key in fees table. In other words, Rollno, which is a primary key in parent (student) table, is referred to as a foreign key in child (fees) table. This means that we cannot insert that value in Rollno field for fees table whose corresponding value is not present in the Rollno field of student table. Thus, it enforces the concept of **Referential Integrity**.

CTM: A table may have multiple Foreign keys. Foreign key may have repeated values.

10.13.3 Viewing Constraints, Viewing the Columns Associated with Constraints

Once we have created a table, its structure and contents can be viewed using DESC command. With this command, we can also see the constraints, if any, associated with the table structure. Thus, with DESC command, table structure as well as constraints gets displayed. A constraint is displayed beside the column name on which it is applicable using either of the following commands:

```
mysql>DESC student;
OR
mysql> SHOW CREATE TABLE student;
```

10.14 SQL QUERY PROCESSING

After creating the database, the table is created and the data is stored into it. Now, it's time to perform query processing on already created tables to retrieve and view the data on the screen. Retrieving information from the tables is done mainly using the SELECT command. The SQL SELECT statement is used to fetch data from one or more database tables. It is used to select rows and columns from a database/relation.

10.14.1 SQL SELECT Statement

This command can perform selection as well as projection. It is the most extensively used SQL command. The SELECT statement can be used to retrieve a subset of rows or columns from one or more tables present in a database.

Resultant table: student

Name	Gender
Raj Kumar	M
Deep Singh	M
Ankit Sharma	M
Radhika Gupta	F
Payal Goel	F
Diksha Sharma	F
Gurpreet Kaur	F
Akshay Dureja	M
Shreya Anand	F
Prateek Mittal	M

1. Selection

This capability of SQL returns the tuples from a relation with all the attributes.

Syntax: SELECT <column_name1> [, <column_name2>...]
FROM <table_name>;
OR SELECT <what_to_select>
FROM <which_table>
WHERE <conditions_to_satisfy>;

For example,

```
SELECT Name, Gender FROM student;
```

The above command displays only the name and gender attributes from the student table.

2. Projection: Selecting Specific Rows—WHERE Clause

This is the capability of SQL to return only specific attributes in the relation. Use of where clause is required when specific tuples are to be fetched or manipulated. To select all the columns from a table, the asterisk (*) can be used.

For example,

```
SELECT * FROM student;
```

The above command will display all the tuples (rows) from the relation student.

CTM: The asterisk (*) means "All". SELECT * means displaying all the columns from a relation.

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no
1.	Raj Kumar	M	93	17-Nov-2000	NULL
2.	Deep Singh	M	98	22-Aug-1996	NULL
3.	Ankit Sharma	M	76	02-Feb-2000	NULL
4.	Radhika Gupta	F	78	03-Dec-1999	NULL
5.	Payal Goel	F	82	21-April-1998	NULL
6.	Diksha Sharma	F	80	17-Dec-1999	NULL
7.	Gurpreet Kaur	F	65	04-Jan-2000	NULL
8.	Akshay Dureja	M	90	05-May-1997	NULL
9.	Shreya Anand	F	70	08-Oct-1999	NULL
10.	Prateek Mittal	M	75	25-Dec-2000	NULL

10 rows in a set (0.02 sec)

The above command displays all the rows of all the columns, according to the column-list defined in the table structure. The salient features of SQL SELECT statement are as follows:

- SELECT command displays the columns of the table in the same order in which they are selected from the table.
- In order to retrieve all the columns in the column-list from a table using SELECT command, asterisk (*) is used and the columns are displayed in the same order in which they are stored in the table.
- All the statements (inclusive of SELECT statement) in SQL are terminated with a semicolon (;). Use of semicolon is dependent on the version in use.

Using WHERE clause

```
SELECT * FROM student WHERE Rollno<=8;
```

The above command shall display only those records whose Rollno is less than or equal to 8.

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no
1.	Raj Kumar	M	93	17-Nov-2000	NULL
2.	Deep Singh	M	98	22-Aug-1996	NULL
3.	Ankit Sharma	M	76	02-Feb-2000	NULL
4.	Radhika Gupta	F	78	03-Dec-1999	NULL
5.	Payal Goel	F	82	21-April-1998	NULL
6.	Diksha Sharma	F	80	17-Dec-1999	NULL
7.	Gurpreet Kaur	F	65	04-Jan-2000	NULL
8.	Akshay Dureja	M	90	05-May-1997	NULL

8 rows in a set (0.02 sec)

When a WHERE clause is used with a SELECT statement, the SQL query processor goes through the entire table, one row/record at a time, and checks each row to determine whether the condition specified is true with respect to that row or not. If it evaluates to True, the corresponding row is selected, retrieved and displayed, else it returns an empty set (*i.e.*, no data found).

3. Recording Columns while Displaying Query Results

While displaying the result for a query, the order of the columns to be displayed can be changed according to the user's requirement. But this is done only for display purpose and no actual (physical) rearrangement of the columns is done.

For example,

```
SELECT Name, Rollno, DOB, Marks from student;
```

After executing the above statement, the column shall be displayed in the changed order and Name shall be displayed as the first column, Rollno as the second column, DOB as the third column and Marks as the fourth column respectively.

Resultant table: student

Name	Rollno	DOB	Marks
Raj Kumar	1.	17-Nov-2000	93
Deep Singh	2.	22-Aug-1996	98
Ankit Sharma	3.	02-Feb-2000	76
Radhika Gupta	4.	03-Dec-1999	78
Payal Goel	5.	21-April-1998	82
Diksha Sharma	6.	17-Dec-1999	80
Gurpreet Kaur	7.	04-Jan-2000	65
Akshay Dureja	8.	05-May-1997	90
Shreya Anand	9.	08-Oct-1999	70
Prateek Mittal	10.	25-Dec-2000	75

10 rows in a set (0.02 sec)

CTM: The order in which the columns are displayed using the SELECT command is in accordance with the order in which they are actually stored in the table.

4. Eliminating Duplicate/Redundant Data—DISTINCT clause

DISTINCT clause is used to remove duplicate rows from the results of a SELECT statement. It is used to retrieve only unique values for a column in the table. The DISTINCT keyword can be used only once with a given SELECT statement.

Syntax:

SELECT DISTINCT <column_name> from <*table_name*>;

For example,

Suppose we have added a new column Stream to the table student:

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	Stream
1.	Raj Kumar	M	93	17-Nov-2000	9586774748	Science
2.	Deep Singh	M	98	22-Aug-1996	8988886577	Commerce
3.	Ankit Sharma	M	76	02-Feb-2000	NULL	Science
4.	Radhika Gupta	F	78	03-Dec-1999	9818675444	Humanities
5.	Payal Goel	F	82	21-April-1998	9845639990	Vocational
6.	Diksha Sharma	F	80	17-Dec-1999	9897666650	Humanities
7.	Gurpreet Kaur	F	65	04-Jan-2000	7560567890	Science
8.	Akshay Dureja	M	90	05-May-1997	9560567890	Commerce
9.	Shreya Anand	F	70	08-Oct-1999	NULL	Vocational
10.	Prateek Mittal	M	75	25-Dec-2000	9999967543	Science

10 rows in a set (0.02 sec)

With reference to the above table, if we write the SELECT statement as:

SELECT Stream from student;

this statement shall return all the tuples for field Stream from table student. It will return duplicate values also. Thus, in order to remove these duplicate values, DISTINCT clause is used.

Now, we write a query for displaying the distinct contents on the basis of field stream from student table:

Resultant table: student

Stream
Science
Commerce
Science
Humanities
Vocational
Humanities
Science
Commerce
Vocational
Science

Science displayed 4 times

10 rows in a set (0.02 sec)

For example,

SELECT **DISTINCT** Stream from student;

Resultant table: student

Stream
Science
Commerce
Humanities
Vocational

Science displayed only once

4 rows in a set (0.02 sec)

10.15 SQL OPERATORS

While working with SELECT statement using WHERE clause, condition-based query processing is carried out using four types of SQL operators:

(a) Arithmetic Operators

(b) Relational Operators

(c) Logical Operators

(d) Special Operators

Table 10.2: SQL Operators and their Functions

OPERATOR/FUNCTION	DESCRIPTION
COMPARISON OPERATORS	
=, >, <, >=, <=, <>	Used in Conditional expressions.
LOGICAL OPERATORS	
AND/OR/NOT	Used in Conditional expressions.
SPECIAL OPERATORS	Used in Conditional expressions.
BETWEEN	Checks whether an attribute value is within a range.
IS NULL	Checks whether an attribute value is null.
LIKE	Checks whether an attribute matches a given string pattern.
IN	Checks whether an attribute value matches any value with a given list.
DISTINCT	Permits only unique values. Eliminates duplicate ones.
AGGREGATE FUNCTIONS	Used with SELECT to return mathematical result/values on the basis of the operation performed on the columns.
COUNT	Returns the total number of records with non-null values for a given column.
MIN	Returns the minimum/lowest attribute value found in a given column.
MAX	Returns the maximum/highest attribute value found in a given column.
SUM	Returns the sum of all the values for a given column.
AVG	Returns the average of all the values for a given column.

(a) Arithmetic Operators

Arithmetic operators are used to perform simple arithmetic operations like addition (+), subtraction (–), multiplication (*), division (/) and modulus (%). These operators are used with conditional expressions and for performing simple mathematical calculations. The arithmetic operators with SELECT command are used to retrieve rows computed with or without reference to any table.

For example,

```
mysql> SELECT 5 + 10;
```

OR

```
mysql> SELECT 5 + 10 FROM DUAL;
```

The above statement shall evaluate the expression 5 + 10 and returns the value 15 as the result.

```
+-----------+
| 5 + 10    |
+-----------+
| 15        |
+-----------+
```

```
+-------------+-----------+
| SIN(PI()/4) | (4+1)*5   |
+-------------+-----------+
| 0.707107    |        25 |
+-------------+-----------+
```

mysql> SELECT SIN(PI()/4), (4+1)*5;

```
mysql> SELECT 5 * 4 FROM DUAL;
+-----------+
| 5 * 4     |
+-----------+
| 20        |
+-----------+
```

```
mysql> SELECT 47 % 5 FROM DUAL;
+-----------+
| 47 % 5    |
+-----------+
|         2 |
+-----------+
```

The modulus (%) operator returns the remainder as the answer after performing the division operation. Hence, the above statement 47 % 5 shall return the value 2 as the ouput.

POINT TO REMEMBER

In the above statement, DUAL is the default table in MySQL. It is a one-row, one-column dummy table.

Evaluating Scalar expression with SELECT statement

MySQL permits calculations on the contents of the columns and then displays the calculated result using SELECT statement. We can write scalar expression and constant values for the selected columns. If we are taking NULL value in the expression, it shall result in NULL only. Along with NULL, arithmetic operators can be used while evaluating scalar expressions.

For example, **mysql>** SELECT Rollno, Name, Marks + 10 FROM student;

The above command, on execution, shall increment the value of all the rows of the field Marks by 10 and shall display the Rollno, Name and Marks for all the students increased by 10.

Resultant table: student

Rollno	Name	Marks + 10
1.	Raj Kumar	103
2.	Deep Singh	108
3.	Ankit Sharma	86
4.	Radhika Gupta	88
5.	Payal Goel	92
6.	Diksha Sharma	90
7.	Gurpreet Kaur	75
8.	Akshay Dureja	100
9.	Shreya Anand	80
10.	Prateek Mittal	85

10 rows in a set (0.02 sec)

(b) Relational Operators

A relational (comparison) operator is a mathematical symbol which is used to compare two values. It is used to compare two values of the same or compatible data types. Comparison operators are used for conditions where two expressions are required to be compared with each other, which results in either true or false. They are used with WHERE clause.

The following table describes different types of comparison operators in SQL:

OPERATOR	DESCRIPTION
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>, !=	Not equal to

For comparing character data type values, < means earlier in the alphabetical sequence and > means later in the alphabetical sequence.

For example, ***mysql>*** SELECT Rollno, Name, Marks FROM student

WHERE Marks>=90;

The above command shall display the Rollno, Name and Marks of all the students with marks either equal to or greater than 90.

Resultant table: student

Rollno	Name	Marks
1.	Raj Kumar	93
2.	Deep Singh	98
3.	Akshay Dureja	90

3 rows in a set (0.02 sec)

For example, ***mysql>*** SELECT * FROM student

WHERE Stream <> 'Commerce';

The above command shall display the records of all the students who are not from Commerce stream.

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	Stream
1.	Raj Kumar	M	93	17-Nov-2000	9586774748	Science
3.	Ankit Sharma	M	76	02-Feb-2000	8567490078	Science
4.	Radhika Gupta	F	78	03-Dec-1999	9818675444	Humanities
5.	Payal Goel	F	82	21-April-1998	9845639990	Vocational
6.	Diksha Sharma	F	80	17-Dec-1999	9897666650	Humanities
7.	Gurpreet Kaur	F	65	04-Jan-2000	7560875609	Science
9.	Shreya Anand	F	70	08-Oct-1999	8876543988	Vocational
10.	Prateek Mittal	M	75	25-Dec-2000	9999967543	Science

8 rows in a set (0.02 sec)

Thus, while using relational operators in a WHERE clause with a SELECT statement, the database program goes through the entire table checking each record one by one and compares it with the condition specified. If it is true, the corresponding row is selected for display, otherwise ignored.

CTM: While comparing character, date and time data using relational operators, it should be enclosed in single quotation marks.

(c) Logical Operators

The SQL logical operators are the operators used to combine multiple conditions to narrow data selected and displayed on the basis of the condition specified in an SQL statement. Logical operators are also known as Boolean operators. The three logical operators in SQL are—AND, OR and NOT operators. Out of these, AND and OR operators are termed as Conjunctive operators since they combine two or more conditions. The AND and OR operators are used to filter records based on more than one condition.

These operators provide a means to make multiple comparisons with different operators in the same SQL statement.

CTM: The order of precedence for logical operators (AND, OR, NOT operator) is NOT(!), AND(&&) and OR(||).

1. AND operator

The AND operator displays a record and returns a true value if all the conditions (usually two conditions) specified in the WHERE clause are true.

Condition 1	Condition 2	Result (AND operation)
True	True	True
True	False	False
False	True	False
False	False	False

As shown in the table, when both condition 1 and condition 2 are true, only then is the result true. If either of them is false, the result becomes false.

For example, to list the details of all the students who have secured more than 80 marks and are male.

```
mysql> SELECT * FROM student
        WHERE Marks > 80 and Gender= 'M';
```

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	Stream
1.	Raj Kumar	M	93	17-Nov-2000	9586774748	Science
2.	Deep Singh	M	98	22-Aug-2000	8988886577	Commerce
8.	Akshay Dureja	M	90	05-May-1997	9560567890	Commerce

3 rows in a set (0.02 sec)

2. OR operator

The OR operator displays a record and returns a true value if either of the conditions (usually two conditions) specified in the WHERE clause is true.

Condition 1	Condition 2	Result (OR operation)
True	True	True
True	False	True
False	True	True
False	False	False

As shown in the table, when either condition 1 or condition 2 is true, the result is true. If both of them are false, only then does the result become false.

For example, to display the roll number, name and stream of all the students who are in either Science or Commerce stream,

```
mysql> SELECT Rollno, Name, Stream FROM student
        WHERE Stream= 'Science' or Stream= 'Commerce';
```

Resultant table: student

Rollno	Name	Stream
1.	Raj Kumar	Science
2.	Deep Singh	Commerce
3.	Ankit Sharma	Science
7.	Gurpreet Kaur	Science
8.	Akshay Dureja	Commerce
10.	Prateek Mittal	Science

6 rows in a set (0.02 sec)

3. NOT operator

NOT operator is also termed as a negation operator. Unlike the other two operators, this operator takes only one condition and gives the reverse of it as the result. It returns a false value if the condition holds true and *vice versa*.

The NOT operator displays a record and returns a true value if the condition specified in the WHERE clause is false.

Condition	Result (NOT operation)
True	False
False	True

As shown in the table, when the condition is true, the result is false. If the condition is false, then the result becomes true.

For example, to display the name and marks of all the students who are not in the vocational stream.

```
mysql> SELECT Name, Marks FROM student
        WHERE NOT (Stream = 'Vocational');
```

Resultant table: student

Name	Marks
Raj Kumar	93
Deep Singh	98
Ankit Sharma	76
Radhika Gupta	78
Diksha Sharma	80
Gurpreet Kaur	65
Akshay Dureja	90
Prateek Mittal	75

8 rows in a set (0.02 sec)

10.16 SQL ALIASES

SQL aliases are used to give an alternate name, *i.e.*, a temporary name to a database table or a column in a table. We can rename a table or a column temporarily by giving another name called alias name which leads to a temporary change (renaming) and does not change the actual name in the database.

Aliases can be used when:

- more than one table is involved in a query
- functions are used in the query
- column names are big or not very readable
- two or more columns are combined together

Using column alias name, we can give different name(s) to column(s) for display (output) purpose only. They are created to make column names more readable. SQL aliases can be used both for tables as well as columns.

- **COLUMN ALIASES** are used to make column headings in the query result set easier to read.
- **TABLE ALIASES** are used to shorten a table name by giving an easy alternate name, making it easier to read or when performing a self join (*i.e.*, listing the same table more than once in the FROM clause).

Syntax for table alias:

```
SELECT <columnname1>, <columnname2>.....
FROM <table_name> AS <alias_name>;
WHERE [<condition>];
```

Syntax for column alias:

```
SELECT <column_name> AS <"alias_name">
FROM <table_name>
WHERE [<condition>];
```

For example,

```
SELECT Name AS "Student_name", DOB AS "Date_of_birth"
FROM student;
```

Alias name for fields, Name and DOB

Table: student

Student_name	Date_of_birth
Raj Kumar	17-Nov-2000
Deep Singh	22-Aug-1996
Ankit Sharma	02-Feb-2000
Radhika Gupta	03-Dec-1999
Payal Goel	21-April-1998
Diksha Sharma	17-Dec-1999
Gurpreet Kaur	04-Jan-2000
Akshay Dureja	05-May-1997
Shreya Anand	08-Oct-1999
Prateek Mittal	25-Dec-2000

10 rows in a set (0.02 sec)

POINTS TO REMEMBER

- If the *alias_name* contains spaces, you must enclose the *alias_name* in quotes.
- It is acceptable to use spaces when you are aliasing a column name. However, it is not generally a good practice to use spaces when you are aliasing a table name.
- The *alias_name* is only valid within the scope of the SQL statement.

➢ **Alias name can be given to a mathematical expression also:**

For example,

```
SELECT 22/7 as PI;
```

Output:

```
+------------+
|     PI     |
+------------+
|  3.1429    |
+------------+
```

10.17 PUTTING TEXT IN THE QUERY OUTPUT

In order to get an organized output from a SELECT query, we can include some user-defined columns at runtime. These columns are displayed with a valid text, symbols and comments in the output only. These included columns will appear as column heads along with the contents for that column.

This makes the query output more presentable by giving a formatted output.

For example,

```
SELECT Rollno, Name 'was born on' DOB
FROM student;
```

Text to be displayed

The above command, on execution, shall display the text "was born on" with every record (tuple) of the table.

Resultant table: student

Rollno	Name	was born on	DOB
1.	Raj Kumar	was born on	17-Nov-2000
2.	Deep Singh	was born on	22-Aug-1996
3.	Ankit Sharma	was born on	02-Feb-2000
4.	Radhika Gupta	was born on	03-Dec-1999
5.	Payal Goel	was born on	21-April-1998
6.	Diksha Sharma	was born on	17-Dec-1999
7.	Gurpreet Kaur	was born on	04-Jan-2000
8.	Akshay Dureja	was born on	05-May-1997
9.	Shreya Anand	was born on	08-Oct-1999
10.	Prateek Mittal	was born on	25-Dec-2000

10 rows in a set (0.02 sec)

10.18 SQL SPECIAL OPERATORS

Apart from several standard library functions discussed earlier, there are some special operators in SQL that perform specific functions.

10.18.1 Conditions Based on a Range—BETWEEN...AND

SQL provides a BETWEEN operator that defines a range of values within which the column value must fall for the condition to become true. The range includes both the lower and upper value. The values can be numbers, text or dates.

Syntax for BETWEEN:

```
mysql>SELECT <column_name(s)>
FROM <table_name>
WHERE <column_name> BETWEEN <value1> AND <value2>;
```

For example,

```
mysql> SELECT Rollno, Name, Marks FROM student WHERE Marks BETWEEN 80 AND 100;
```

The above command displays Rollno, Name along with Marks of those students whose Marks lie in the range of 80 to 100 (both 80 and 100 are included in the range).

Resultant table: student

Rollno	Name	Marks
1.	Raj Kumar	93
2.	Deep Singh	98
5.	Payal Goel	82
6.	Diksha Sharma	80
8.	Akshay Dureja	90

5 rows in a set (0.02 sec)

NOT BETWEEN

The NOT BETWEEN operator works opposite to the BETWEEN operator. It retrieves the rows which do not satisfy the BETWEEN condition.

For example,

mysql> SELECT Rollno, Name, Marks FROM student WHERE Marks NOT BETWEEN 80 AND 100;

Resultant table: student

Rollno	Name	Marks
3.	Ankit Sharma	76
4.	Radhika Gupta	78
9.	Shreya Anand	70
10.	Prateek Mittal	75

4 rows in a set (0.02 sec)

10.18.2 Conditions Based on a List—IN

To specify a list of values, IN operator is used. This operator selects values that match any value in the given list. The SQL IN condition is used to help reduce the need for multiple OR **conditions** in a SELECT statement.

Syntax for IN:

SELECT *<column_name(s)>*

FROM *<table_name>*

WHERE *<column_name>* IN *(value1,value2,...);*

For example,

mysql> SELECT * FROM student WHERE Stream IN ('Science', 'Commerce', 'Humanities');

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	Stream
1.	Raj Kumar	M	93	17-Nov-2000	9586774748	Science
2.	Deep Singh	M	98	22-Aug-1996	8988886577	Commerce
3.	Ankit Sharma	M	76	02-Feb-2000	NULL	Science
4.	Radhika Gupta	F	78	03-Dec-1999	9818675444	Humanities
6.	Diksha Sharma	F	80	17-Dec-1999	9897666650	Humanities
7.	Gurpreet Kaur	F	65	04-Jan-2000	7560875609	Science
8.	Akshay Dureja	M	90	05-May-1997	9560567890	Commerce
10.	Prateek Mittal	M	75	25-Dec-2000	9999967543	Science

8 rows in a set (0.02 sec)

The above command displays all those records whose Stream is either Science or Commerce or Humanities.

NOT IN

The NOT IN operator works opposite to IN operator. It matches, finds and returns the rows that do not match the list.

For example,

mysql> SELECT * FROM student WHERE Stream NOT IN ('Science', 'Commerce', 'Humanities');

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	Stream
5.	Payal Goel	F	82	21-April-1998	9845639990	Vocational
9.	Shreya Anand	F	70	08-Oct-1999	NULL	Vocational

2 rows in a set (0.02 sec)

10.18.3 Conditions Based on Pattern—LIKE

The LIKE operator is used to search for a specified pattern in a column. This operator is used with the columns of type CHAR. The LIKE operator searches the column to find if the part of this column matches the string specified in the parentheses after the LIKE operator in the command.

Conditions Based on Pattern—WILD CARD CHARACTERS

The SQL LIKE condition allows you to use wild cards to perform pattern matching. SQL provides two wild card characters that are used while comparing the strings with LIKE operator.

a. **Percent(%)** Matches any string

b. **Underscore(_)** Matches any one character

Syntax for LIKE:

SELECT *<column_name(s)>*

FROM *<table_name>*

WHERE *<column_name>* LIKE *<pattern>;*

For example,

- ***mysql>*** SELECT * FROM student WHERE Name LIKE "D%";

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	Stream
2.	Deep Singh	M	98	22-Aug-1996	8988886577	Commerce
6.	Diksha Sharma	F	80	17-Dec-1999	9897666650	Humanities

2 rows in a set (0.02 sec)

The above command shall display those records where the name begins with character 'D'.

- ***mysql>*** SELECT * FROM student WHERE Name LIKE "%a";

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	Stream
3.	Ankit Sharma	M	76	02-Feb-2000	NULL	Science
4.	Radhika Gupta	F	78	03-Dec-1999	9818675444	Humanities
6.	Diksha Sharma	F	80	17-Dec-1999	9897666650	Humanities
8.	Akshay Dureja	M	90	05-May-1997	9560567890	Commerce

4 rows in a set (0.02 sec)

The above command shall display the records for those students whose name ends with letter 'a'.

- ***mysql>*** SELECT * FROM student WHERE Name LIKE "%e%;

Resultant table: student

Rollno	Name	Gender	Marks	DOB	Mobile_no	Stream
2.	Deep Singh	M	98	22-Aug-1996	8988886577	Commerce
5.	Payal Goel	F	82	21-April-1998	9845639990	Vocational
7.	Gurpreet Kaur	F	65	04-Jan-2000	7560875609	Science
8.	Akshay Dureja	M	90	05-May-1997	9560567890	Commerce
9.	Shreya Anand	F	70	08-Oct-1999	NULL	Vocational
10.	Prateek Mittal	M	75	25-Dec-2000	9999967543	Science

6 rows in a set (0.02 sec)

As the resultant table shows, the above command displays the records of all the students whose Name contains character 'e' in it.

- ***mysql>*** SELECT * FROM student WHERE Name LIKE "_e%;

Resultant table: student

Rollno	Name	DOB
2.	Deep Singh	22-Aug-1996

1 row in a set (0.02 sec)

This command shall display the Rollno, Name and DOB of all the students whose Name contains letter 'e' at the second place.

- *mysql>* SELECT Rollno, Name, Marks, DOB FROM student WHERE Name LIKE "%r__";

Resultant table: student

Rollno	Name	Marks	DOB
3.	Ankit Sharma	76	02-Feb-2000
6.	Diksha Sharma	80	17-Dec-1999

2 rows in a set (0.02 sec)

This command shall display the Rollno, Name, Marks and DOB of all the students whose Name contains letter 'r' from the third last position.

10.19 SORTING IN SQL—ORDER BY

The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. The ORDER BY keyword is used to sort the result-set by one or more fields in a table. This clause sorts the records in the ascending order (ASC) by default. Therefore, in order to sort the records in descending order, DESC keyword is to be used. Sorting using ORDER BY clause can be done on multiple columns, separated by comma.

Syntax for ORDER BY clause:

SELECT <column-list> FROM <table_name>
[WHERE <condition>]
ORDER BY <column_name> [ASC|DESC];

Here, WHERE clause is optional.

For example,

- To display the roll number, name and marks of students on the basis of their marks in the ascending order.

  mysql> SELECT Rollno, Name, Marks
  FROM student
  ORDER BY Name;

Resultant table: student

Rollno	Name	Marks
8.	Akshay Dureja	90
3.	Ankit Sharma	76
2.	Deep Singh	98
6.	Diksha Sharma	80
7.	Gurpreet Kaur	NULL
5.	Payal Goel	82
10.	Prateek Mittal	75
4.	Radhika Gupta	78
1.	Raj Kumar	93
9.	Shreya Anand	70

10 rows in a set (0.02 sec)

- To display the roll number, name and marks of all the students in the descending order of their marks and ascending order of their names.

  mysql> SELECT Rollno, Name, Marks
  FROM student
  ORDER BY Marks DESC, Name;

Resultant table: student

Rollno	Name	Marks
2.	Deep Singh	98
1.	Raj Kumar	93
8.	Akshay Dureja	90
5.	Payal Goel	82
6.	Diksha Sharma	80
4.	Radhika Gupta	78
3.	Ankit Sharma	76
10.	Prateek Mittal	75
9.	Shreya Anand	70
7.	Gurpreet Kaur	NULL

10 rows in a set (0.02 sec)

Sorting on Column Alias

If a column alias is defined for a column, it can be used for displaying rows in ascending or descending order using ORDER BY clause.

For example,

```
SELECT Rollno, Name, Marks AS Marks_obtained
FROM student
ORDER BY Marks_obtained;
```

Alias name

10.20 AGGREGATE FUNCTIONS

Till now we have studied about single row functions which work on a single value. SQL also provides multiple row functions, which work on multiple values. So, we can apply SELECT query on a group of records rather than the entire table. Therefore, these functions are called aggregate functions or group functions.

Generally, the following aggregate functions are applied on groups as described below:

Table 10.4: Aggregate Functions in SQL

S.No.	Function	Description/Purpose
1.	MAX()	Returns the maximum/highest value among the values in the given column/expression.
2.	MIN()	Returns the minimum/lowest value among the values in the given column/expression.
3.	SUM()	Returns the sum of the values under the specified column/expression.
4.	AVG()	Returns the average of the values under the specified column/expression.
5.	COUNT()	Returns the total number of values/records under the specified column/expression.

Consider a table Employee (Employee code, employee name, job, salary and city) with the following structure:

Ecode	Ename	Salary	Job	City
E1	Ritu Jain	5000	Manager	Delhi
E2	Vikas Verma	4500	Executive	Jaipur
E3	Rajat Chaudhary	6000	Clerk	Kanpur
E4	Leena Arora	7200	Manager	Bangalore
E5	Shikha Sharma	8000	Accountant	Kanpur

1. MAX()

MAX() function is used to find the highest value among the given set of values of any column or expression based on column. MAX() takes one argument which can be either a column name or any valid expression involving a particular column from the table.

For example,

mysql> Select MAX(Salary) from EMPLOYEE;

Output:

```
+---------------+
| MAX(Salary)|
+---------------+
| 8000          |
+---------------+
```

This command on execution shall return the maximum value from the specified column (Salary) of the table Employee, which is 8000.

2. MIN()

MIN() function is used to find the lowest value among the given set of values of any column or expression based on column. MIN() takes one argument which can be either a column name or any valid expression involving a particular column from the table.

For example,

mysql> Select MIN(Salary) from EMPLOYEE;

Output:

```
+---------------+
| MIN(Salary)|
+---------------+
| 4500          |
+---------------+
```

This command on execution shall return the minimum value from the specified column (Salary) of the table Employee, which is 4500.

3. SUM()

SUM() function is used to find the total value of any column or expression based on a column. It accepts the entire range of values as an argument, which is to be summed up on the basis of a particular column, or an expression containing that column name. The SUM() function always takes argument of integer type only. Sums of String and Date type data are not defined.

For example,

mysql> Select SUM(Salary) from EMPLOYEE;

```
+---------------+
| SUM(Salary)|
+---------------+
| 30700         |
+---------------+
```

This command on execution shall return the total of the salaries of all the employees from the specified column (Salary) of the table Employee, which is 30700.

4. AVG()

AVG() function is used to find the average value of any column or expression based on a column. Like sum(), avg(), it also accepts the entire range of values of a particular column to be taken average of, or even a valid expression based on this column name. Like SUM() function, the AVG() function always takes argument of integer type only. Average of String and Date type data are not defined.

For example,

mysql> Select AVG(Salary) from EMPLOYEE;

```
+---------------+
| AVG(Salary) |
+---------------+
| 6140          |
+---------------+
```

This command on execution shall return the average of the salaries of all the employees from the specified column (Salary) of the table Employee, which is 6140.

5. COUNT()

COUNT() function is used to count the number of values in a column. COUNT() takes one argument which can be any column name, or an expression based on a column, or an asterisk (*). When the argument is a column name or an expression based on column, COUNT() returns the number of non-NULL values in that column. If the argument is asterisk (*), then COUNT() counts the total number of records/rows satisfying the condition, if any, in the table.

For example,

1. **mysql> Select COUNT(*) from EMPLOYEE;**

```
+---------------+
| COUNT(*)    |
+---------------+
| 5             |
+---------------+
```

This command on execution shall return the total number of records in the table Employee, which is 5.

2. **mysql> Select COUNT(DISTINCT City) from EMPLOYEE;**

```
+----------------------------+
| COUNT(DISTINCT City) |
+----------------------------+
| 4                          |
+----------------------------+
```

This command on execution shall return the total number of records on the basis of city with no duplicate values, *i.e.*, Kanpur is counted only once; the second occurrence is ignored by mySQL because of the DISTINCT clause. Thus, the output will be 4 instead of 5.

Aggregate Functions & NULL Values

Consider the following table Employee with NULL values against the Salary field for some employees.

Employee

Ecode	Ename	Salary	Job	City
E1	Ritu Jain	NULL	Manager	Delhi
E2	Vikas Verma	4500	Executive	Jaipur
E3	Rajat Chaudhary	6000	Clerk	Kanpur
E4	Leena Arora	NULL	Manager	Bangalore
E5	Shikha Sharma	8000	Accountant	Kanpur

None of the aggregate functions takes NULL into consideration. NULL values are simply ignored by all the aggregate functions as clearly shown in the examples given below:

mysql> Select Sum(Salary) from Employee;

Output: 18500

mysql> Select Min(Salary) from Employee;

Output: 4500 (NULL values are not considered.)

mysql> Select Max(Salary) from Employee;

Output: 8000 (NULL values are not ignored.)

mysql> Select Count(Salary) from Employee;

Output: 3 (NULL values are not ignored.)

mysql> Select Avg(Salary) from Employee;

Output: 6166.66 (It will be calculated as 18500/3, *i.e*, sum/total no. of records (which are 3 after ignoring NULL values).)

mysql> Select Count(*) from Employee;

Output: 5

mysql> Select Count(Ecode) from Employee;

Output: 5 (No NULL value exists in the column Ecode.)

mysql> Select Count(Salary) from Employee;

Output: 3 (NULL values are ignored while counting the total number of records on the basis of Salary.)

MEMORY BYTES

- Basic/raw facts about something which are not organized are termed as Data. *For example*, details of some students which are not organized.
- Each piece of information about an entity, such as name of a person or address, age or name of a product or the price, is a Field/Data Item.

- A collection of data is referred to as database and a database (management) system is basically a computer-based record-keeping system.
- SQL is a language that is used to create, modify and access a database.
- The various processing capabilities of SQL are: Data Definition Language (DDL), Data Manipulation Language (DML) and Data Control Language (DCL).
- DDL is used to create and delete tables, views or indexes.
- Database systems help reduce data redundancy, data inconsistency and facilitate sharing of data, standardization of data and data security.
- A relational database organizes the data into tables known as relations.
- A group of rows and columns forms a Table.
- The horizontal subset of a Table is known as a Row/Tuple.
- The vertical subset of a Table is known as a Column/Attribute.
- DML is used to modify and update the database.
- DESCRIBE or DESC is used to show the structure of a table.
- The SELECT statement is used to fetch data from one or more database tables.
- SELECT * means display all columns.
- The WHERE clause is used to select specific rows.
- The DESCRIBE statement is used to see the structure of a table.
- We can change the structure of a table, *i.e.*, add, remove or change its column(s) using the ALTER TABLE statement.
- The keyword DISTINCT is used to eliminate redundant data from display.
- Logical operators OR and AND are used to connect relational expressions in WHERE clause.
- Logical operator NOT is used to negate a condition.
- The BETWEEN operator defines a range of values within which the column value must fall for the condition to become true.
- The IN operator selects values that match any value in the given list of values.
- % and _ are two wild card characters. The per cent (%) symbol is used to represent any sequence of zero or more characters. The underscore (_) symbol is used to represent a single character.
- NULL represents a value that is unavailable, unassigned, unknown or inapplicable.
- The results of the SELECT statement can be displayed in the ascending or descending order of a single column or columns using ORDER BY clause.
- DROP DATABASE drops all tables in the database and deletes the database. Once the DROP command is used, then we cannot use that database. So, we should be careful with this command.
- The CREATE statement is used to create a table in MySQL with constraint. A constraint is a restriction on the behaviour of a variable.
- INSERT query is used for inserting new rows or data into an existing table.
- The ORDER BY clause is used to sort and display the records of a table on the basis of a specific column/field in either ascending or descending order.

SOLVED QUESTIONS

Very Short Answer Questions

1. What is DBMS?

Ans. Database is managed by a software package known as Database Management System (DBMS).

2. List all possible attributes for a student entity.

Ans. Name, Rollno, Marks, Address, Phone No., Date of birth.

3. If R1 is a relation with 5 rows and R2 is a relation with 3 rows, how many rows will the Cartesian product of R1 and R2 have?

Ans. 15

4. What is a relation? What is the difference between a tuple and an attribute?

Ans. A relation is like a table in which data is arranged in the form of rows and columns.

- The rows of a table are called tuples.
- The columns of a table are called attributes.

5. What is data type?

Ans. The attribute of a field that determines the kind of data the field can contain is called data type.

6. What is a relation?

Ans. A relation is a two-dimensional table consisting of rows and columns as records and attributes respectively.

7. What do you understand by Degree and Cardinality of a table?

Ans. Degree of a table is the total number of attributes.
Cardinality of a table is the total number of rows.

8. What do you understand by primary key and candidate key?

Ans. An attribute or a set of attributes which is used to identify a tuple uniquely is known as primary key. If a table has more than one such attribute which identifies a tuple uniquely, then all such attributes are known as candidate keys.

9. What do you understand by the term "Primary Key" and "Degree of relation" in relational database?

Ans. *Primary Key:* The attribute (column) or set of attributes (columns) which is used to identify a tuple/row uniquely is known as primary key.
Degree of relation: Number of attributes or columns in a table forms cardinality of a relation.

10. What is a record?

Ans. A record is an arrow on a datasheet and is a set of values defined by fields.

11. What do you mean by data redundancy?

Ans. Data redundancy means duplication of data. Duplicate data is stored at different locations which violates the integrity of the database and causes wastage of storage space.

12. How does database management system ensure data security and privacy?

Ans. A database management system ensures data security and privacy by providing authorized access privileges only to the authorized person. Grant and revoke permission can be given only to authorized persons of an organization to gain access to sensitive data.

13. What do you mean by data security?

Ans. Data security refers to protection of data against accidental or intentional disclosure to unauthorized persons, or unauthorized modification or destruction.

14. What is data integrity?

Ans. Data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life cycle.

15. What do you mean by relational database?

Ans. A relational database is a collection of data items organized as a set of tables from which data can be retrieved easily and efficiently.

16. Differentiate between primary key and alternate key.

Ans. *Primary key:* Every table naturally includes one or more fields that comprise a primary key which differentiates one record from the other.
Alternate key: There may be two or more attributes or combinations of attributes which uniquely identify an instance of an entity set. In such a case, we must decide which of the candidate keys will be used as the primary key. The remaining candidate keys would be considered as alternate keys.

17. What do you understand by candidate keys in a table? Give a suitable example of candidate key from a table containing some meaningful data.

Ans. A table may have more than one such attribute/group of attributes that identifies a tuple uniquely. All such attribute(s) are known as candidate keys.

For example,

Table: Item

Item_no	Item_name	Qty
101	Pen	560
102	Pencil	780
104	CD	450
109	Long file	350
105	Eraser	250
103	Duster	128

In the above table, Item_no and Item_name are treated as candidate keys.

18. What is SQL?

Ans. Structured Query Language (SQL) is a language used for accessing databases.

19. Define the following terms: Field, Record, Table

Ans. Field: A field is the smallest unit of a table which is also known as column. Columns are called attributes in a table. *For example*, Employee Name, Employee ID, etc.

Record: A record is the collection of values / fields of a specific entity. *For example*, record of an employee that consists of Employee name, Salary, etc.

Table: A table is called a relation. It is a collection of records of a specific type of data. *For example*, employee table, salary table, etc., that can consist of the records of employees and record of salary respectively.

20. What are DDL and DML statements?

Ans. DDL statements are used for creating or deleting tables, views, etc. DML statements are used for manipulating values for records in a table.

21. What is a base table?

Ans. A table from which the values can be derived for other table is known as a base table. *For example*, in case of views, the values are extracted from the base table on which the view depends.

22. What is NULL value?

Ans. A NULL value in a table is a value in a field which is blank, which means a field with a NULL value is a field with no value; not even zero is entered.

23. What is a NOT NULL constraint?

Ans. If you do not want a column to have a NULL value, then you need to define a constraint on this column specifying that NULL is now not allowed for that column.

Short Answer Questions

1. What do you mean by Primary Key? Give a suitable example of Primary Key from a table containing some meaningful data.

Ans. An attribute or a set of attributes which is used to identify a tuple (row) uniquely is known as Primary Key.

Table: Students

Admission_No	First Name	Last Name	DOB
27354	Jatin	Kumar	05-02-1998
25350	Mona	Sinha	24-09-2004
26385	George	Moun	19-05-1997
16238	Mukesh	Kumar	24-09-2004

Admission_No. is the Primary Key because this column will contain unique data for each record.

2. Write a query that displays city, salesman name, code and commission from salesman table.

Ans. SELECT city, salesman_name, code, commission from salesman;

3. Write a query that selects all orders except those with zero or NULL in the amount field from table orders.

Ans. SELECT * from orders where amount IS NOT NULL;

4. Write a command that deletes all orders for customer SOHAN from table customer.

Ans. DELETE from customer where customer_name is LIKE 'SOHAN';

5. Differentiate between DROP and DELETE command.

Ans. DROP command is used to drop a table with all records stored in it whereas delete command is used to delete all records or some of the records from a table without deleting the table.

6. How can you add a new column or a constraint in a table?

Ans. If you want to add a new column in an existing table, ALTER command is used. *For example*, to add a column bonus in a table emp, the statement will be given as:

```
ALTER table emp
ADD (bonus Integer);
```

7. Can you add more than one column in a table by using the ALTER TABLE command?

Ans. Yes, we can add more than one column by using the ALTER TABLE command. Multiple column names separated by comma are given along with ADD clause in ALTER TABLE command. *For example*, to add city and pin code in a table employee, the command can be given as:

```
ALTER table employee
ADD (city CHAR(30), PINCODE INTEGER);
```

8. How can you eliminate duplicate records in a table with select query?

Ans. The DISTINCT clause is used with SELECT statement to hide duplicate records in a table. *For example*, to display cities from table suppliers.

```
SELECT DISTINCT city FROM suppliers;
```

9. How can you DROP a UNIQUE Constraint?

Ans. To drop a UNIQUE constraint, use the following SQL statement:

```
ALTER TABLE SUPPLIER
DROP CONSTRAINT chkcommission;
```

Here, the constraint chkcommission is deleted from table supplier.

10. Define a Foreign Key.

Ans. A foreign key is a key which is used to link two tables together. It is also called a referencing key. Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table. The relationship between two tables matches the Primary Key in one of the tables with a Foreign Key in the second table. If a table has a primary key defined on any field(s), then you cannot have two records having the same value of that field(s).

11. Define the various SQL Constraints.

Ans. Constraints are the rules enforced on data or columns on a table. These are used to restrict the values that can be inserted in a table. This ensures accuracy and reliability of the data in the database.

Following are the most commonly-used constraints available in SQL:

- NOT NULL Constraint: Ensures that a column cannot have NULL value.
- DEFAULT Constraint: Provides a default value for a column when no value is specified.
- UNIQUE Constraint: Ensures that all values in a column are unique. There should not be any redundant value in a column which is being restricted.
- PRIMARY Key: Uniquely identifies each row/record in a database table.
- FOREIGN Key: Uniquely identifies a row/record in any other database table.
- CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions. *For example*, to restrict the salary column that it should contain salary more than ₹ 10,000.

12. Consider the following tables: COMPANY and MODEL.

Table: Company

Comp_ID	CompName	CompHO	ContactPerson
1	Titan	Okhla	C.B. Ajit
3	Ajanta	Najafgarh	R. Mehta
2	Maxima	Shahdara	B. Kohli
4	Seiko	Okhla	R. Chadha
5	Ricoh	Shahdara	J. Kishore

Note:

- Comp_ID is the Primary Key.

Table: Model

Model_ID	Comp_ID	Cost	DateOfManufacture
T020	1	2000	2010-05-12
M032	4	7000	2009-04-15
M059	2	800	2009-09-23
A167	3	1200	2011-01-12
T024	1	1300	2009-10-14

Note:

- Model_ID is the Primary Key.
- Comp_ID is the Foreign Key referencing Comp_ID of Company table.

Write SQL commands for queries (i) to (iii) and output for (iv) and (v).

(i) To display the details of all the models in the Model table in ascending order of DateOfManufacture.

(ii) To display the details of those models manufactured in 2011 and whose Cost is below 2000.

(iii) To decrease the cost of all the models in Model table by 15%.

(iv) Select COUNT(DISTINCT CompHO) from Company;

(v) Select CompName, contact('Mr.',ContactPerson) from Company where CompName ends with 'a';

Ans. (i) select * from model
order by DateOfManufacture;

(ii) select * from model
where year(DateOfManufacture) = 2011 and cost < 2000;

(iii) Update Model
set Cost = Cost – 0.15*Cost;

(iv) 3

(v) Ajanta Mr. R. Mehta
Maxima Mr. B. Kohli

13. Consider the following two tables: PRODUCT and CLIENT.

Table: Product

P_ID	ProductName	Manufacturer	Price	ExpiryDate
TP01	Talcum Powder	LAK	40	2011-06-26
FW05	Face Wash	ABC	45	2010-12-01
BS01	Bath Soap	ABC	55	2010-09-10
SH06	Shampoo	XYZ	120	2012-04-09
FW12	Face Wash	XYZ	95	2010-08-15

Note:

- P_ID is the Primary Key.

Table: Client

C_ID	ClientName	City	P_ID
1	Cosmetic Shop	Delhi	FW05
6	Total Health	Mumbai	BS01
12	Live Life	Delhi	SH06
15	Pretty One	Delhi	FW05
16	Dreams	Bengaluru	TP01
14	Expressions	Delhi	NULL

Note:

- C_ID is the Primary Key.
- P_ID is the Foreign Key referencing P_ID of Client table.

Write SQL statements for the queries (i) to (iii) and output for (iv) and (v):

(i) To display the ClientName and City of all Mumbai and Delhi-based clients in Client table.

(ii) Increase the price of all the products in Product table by 10%.

(iii) To display the ProductName, Manufacturer, ExpiryDate of all the products that expired on or before '2010-12-31'.

(iv) Select COUNT(DISTINCT Manufacturer) from Product;

(v) Select C_ID, Client_Name, City from Client where City Like 'M%';

Ans. (i) Select ClientName, City from Client
where City = 'Mumbai' or City = 'Delhi';

(ii) Update Product
set Price = Price + 0.10 * Price;

(iii) Select ProductName, Manufacturer, ExpiryDate from Product
where ExpiryDate < = '2010-12-31';

(iv) 3

(v) 6 Total Health Mumbai

14. Consider the following two tables: STATIONERY and CONSUMER.

Table: Stationery

S_ID	StationeryName	Company	Price	StockDate
DP01	Dot Pen	ABC	10	2011-03-31
PL02	Pencil	XYZ	6	2010-01-01
ER05	Eraser	XYZ	7	2010-02-14
PL01	Pencil	CAM	5	2009-01-09
GP02	Gel Pen	ABC	15	2009-03-19

Note:

- S_ID is the Primary Key.

Table: Consumer

C_ID	ConsumerName	Address	P_ID
01	Good Learner	Delhi	PL01
06	Write Well	Mumbai	GP02
12	Topper	Delhi	DP01
15	Write & Draw	Delhi	PL02
16	Motivation	Bengaluru	PL01

Note:

- C_ID is the Primary Key.
- P_ID is the Foreign Key referencing S_ID of Stationery table.

Write SQL statements for the queries (i) to (iii) and output for (iv) and (v):

(i) To display details of all the Stationery items in the Stationery table in descending order of StockDate.

(ii) To display details of that Stationery item whose Company is XYZ and price is below 10.

(iii) To increase the price of all the Stationery items in Stationery table by ₹ 2.

(iv) Select COUNT(DISTINCT Address) from Consumer;

(v) Select StationeryName, price * 3 from Stationery
where Company = 'CAM';

Ans. (i) Select * from Stationery
order by StockDate desc;

(ii) Select * from Stationery
where Company = 'XYZ' and Price < 10;

(iii) Update Stationery
Set Price = Price + 2;

(iv) 3

(v) Pencil 15

15. Consider the following tables: STOCK and DEALER.

Table: Stock

ItemNo	Item	Dcode	Qty	UnitPrice	StockDate
5005	Ball Pen 0.5	102	100	16	2011-03-31
5003	Ball Pen 0.25	102	150	20	2010-01-01
5002	Gel Pen Premium	101	125	14	2010-02-14
5006	Gel Pen Classic	101	200	22	2009-01-09
5001	Eraser Small	102	210	5	2009-03-19
5004	Eraser Big	102	60	10	2010-12-12
5009	Sharpener Classic	103	160	8	2010-01-23

Note:

- ItemNo is the Primary Key.
- Dcode is the Foreign Key referencing Dcode of Dealer table.

Table: Dealer

Dcode	DName
101	Reliable Stationers
103	Class Plastics
104	Fair Deals
102	Clear Deals

Note:

- Dcode is the Primary Key.

Write SQL statements for the queries (i) to (iii) and output for (iv) and (v):

(i) To display details of all the Items in the Stock table in ascending order of StockDate.

(ii) To display details of those Items in Stock table whose Dealer Code(Dcode) is 102 or quantity in Stock(Qty) is more than 100.

(iii) To inset a record in the Stock table with the values:
(5010, 'Pencil HB', 102, 500, 10, '2010-01-26')

(iv) Select COUNT(DISTINCT Dcode) from Stock;

(v) Select Qty * UnitPrice from Stock where ItemNo=5006;

Ans. (i) Select * from Stock
order by StockDate;

(ii) Select * from Stock
where Dcode = 102 or Qty > 100;

(iii) Insert into Stock
values (5010, 'Pencil HB', 102, 500, 10, '2010-01-26');

(iv) 3

(v) 4400

Long Answer Questions

1. (a) Explain the concept of Cartesian product between two tables with the help of an example.

Note: Answer the questions (b) and (c) on the basis of the following tables SHOP and ACCESSORIES.

Table: SHOP

Id	SName	Area
S01	ABC Computronics	CP
S02	All Infotech Media	GK II
S03	Tech Shoppe	CP
S04	Geek Tenco Soft	Nehru Place
S05	Hitech Tech Store	Nehru Place

Table: ACCESSORIES

No	Name	Price	Id
A01	Motherboard	12000	S01
A02	Hard Disk	5000	S01
A03	Keyboard	500	S02
A04	Mouse	300	S01
A05	Motherboard	13000	S02
A06	Keyboard	400	S03
A07	LCD	6000	S04
T08	LCD	5500	S05
T09	Mouse	350	S05
T010	Hard Disk	450	S03

Ans. When you join two or more tables without any condition, it is called Cartesian product or Cross Join.

Example: SELECT * FROM SHOP, ACCESSORIES;

(b) Write the SQL queries:

(i) To display Name and Price of all the Accessories in ascending order of their Price.

(ii) To display Id and SName of all Shops located in Nehru Place.

(iii) To display Minimum and Maximum Price of all the accessories.

Ans. (i) SELECT Name, Price FROM ACCESSORIES ORDER BY Price;

(ii) SELECT Id, SName FROM SHOP WHERE Area='Nehru Place';

(iii) SELECT Name, MAX(Price), MIN(Price) FROM ACCESSORIES;

(c) Write the output of the following SQL commands:

(i) SELECT DISTINCT NAME FROM ACCESSORIES WHERE PRICE>=5000;

(ii) SELECT AREA, COUNT(*) FROM SHOP GROUP BY AREA;

(iii) SELECT COUNT(DISTINCT AREA) FROM SHOP;

Ans. (i) Name
Motherboard
Hard Disk
LCD

(ii) AREA COUNT
CP 2
GK II 1
Nehru Place 2

(iii) COUNT 3

(iv) NAME PRICE
Keyboard 25.00
Motherboard 650.00
Keyboard 20.00
Hard Disk 225.00

2. (a) Define a candidate key with example.

(b) Write SQL queries for (c) to (h) and write the output for the SQL queries mentioned in parts (i1) to (i3) on the basis of tables PRODUCTS and SUPPLIERS.

Table: PRODUCTS

PID	SName	QTY	PRICE	COMPANY	SUPCODE
101	DIGITAL CAMERA 14X	120	12000	RENIX	S01
102	DIGITAL PAD 11i	100	22000	DIGI POP	S02
104	PEN DRIVE 16 GB	500	1100	STOREKING	S01
106	LED SCREEN 32	70	28000	DISPEXPERTS	S02
105	CAR GPS SYSTEM	60	12000	MOVEON	S03

Table: SUPPLIERS

SUPCODE	SNAME	CITY
S01	GET ALL INC	KOLKATA
S03	EASY MARKET CORP	DELHI
S02	DIGI BUSY GROUP	CHENNAI

Ans. (a) A table may have more than one such attribute/group of attributes that identifies a tuple uniquely; all such attribute(s) are known as Candidate Keys.

Table: Item

Ino	Item	QTY
101	Pen	560
102	Pencil	780
104	CD	450
109	Floppy	700
105	Eraser	300
103	Duster	200

Candidate Keys

(c) To display the details of all the products in ascending order of product names (*i.e.*, PNAME).

Ans. SELECT * FROM PRODUCTS ORDER BY PNAME;

(d) To display product name and price of all those products whose price is in the range of 10000 and 15000 (both values inclusive).

Ans. Select PNAME, PRICE FROM PRODUCTS WHERE PRICE>=10000 && PRICE<=15000;

(e) To display the number of products which are supplied by the supplier, *i.e.*, the expected output should be:

S01 2

S02 2

S03 1

Ans. SELECT SUPCODE, COUNT (SUPCODE) FROM PRODUCTS GROUP BY SUPCODE;

(f) To display the price, product name and quantity (*i.e.*, qty) of those products whose quantity is more than 100.

Ans. SELECT PRICE, PNAME, QTY FROM PRODUCTS WHERE QTY>100;

(g) To display the names of those suppliers who are either from DELHI or from CHENNAI.

Ans. SELECT SNAME FROM SUPPLIERS WHERE CITY="DELHI" || CITY="KOLKATA";

(h) To display the name of the companies and the name of the products in descending order of company names.

Ans. SELECT COMPANY, PNAME FROM PRODUCTS ORDER BY COMPANY DESC;

(i) Obtain the outputs of the following SQL queries based on the data given in tables PRODUCTS and SUPPLIERS above.

(i1) SELECT DISTINCT SUPCODE FROM PRODUCTS;

(i2) SELECT MAX(PRICE), MIN(PRICE) FROM PRODUCTS;

(i3) SELECT PRICE*QTY AMOUNT FROM PRODUCTS WHERE PID=104;

Ans. (i1)

DISTINCT SUPCODE
S01
S02
S03

(i2)

MAX (PRICE)	MIN (PRICE)
28000	1100

(i3)

PRICE*QTY
550000

3. (a) Give a suitable example of a table with sample data and illustrate Primary and Alternate Keys in it.

Ans. **Primary Key:** Primary key is a set of one or more fields/columns of a table that uniquely identifies a record in database table. It cannot accept null, duplicate values. Only one Candidate Key can be a Primary Key.

Alternate key: Alternate key is a key that can work as a primary key. Basically it is a candidate key that is currently not a primary key.

Example: In the table given below, AdmissionNo becomes the Alternate Key when we define RegistrationNo as the Primary Key.

Student Registration Table:

RegistrationNo	AdmissionNo	Name	Phone	Gender	DOB
CBSE4554	215647	Mihir Ranjan	9568452325	Male	1992-04-15
CBSE6985	265894	Amita Guha	8456985445	Female	1993-03-24
CBSE5668	458961	Rajesh Singh	9654212440	Male	1992-12-04
CBSE3654	469799	Mohit Patel	7421589652	Male	1992-05-16

Primary Key – RegistrationNo

Alternate Key – AdmissionNo

(b) Consider the following tables CARDEN and CUSTOMER and answer (c) and (d) parts of the question:

Table: CARDEN

Ccode	CarName	Make	Colour	Capacity	Charges
501	A-Star	Suzuki	RED	3	14
503	Indigo	Tata	SILVER	3	12
502	Innova	Toyota	WHITE	7	15
509	SX4	Suzuki	SILVER	4	14
510	C Class	Mercedes	RED	4	35

Table: CUSTOMER

Ccode	CName	Ccode
1001	Hemant Sahu	501
1002	Raj Lal	509
1002	Feroza Shah	503
1004	Ketan Dhal	502

(c) Write SQL commands for the following statements:

(i) To display names of all the silver coloured cars.

(ii) To display the name, make and capacity of cars in descending order of their sitting capacity.

(iii) To display the highest charges at which a vehicle can be hired from CARDEN.

Ans. (i) SELECT CarName FROM carden WHERE Color LIKE 'Silver';

(ii) SELECT CarName,Make,Capacity FROM carden ORDER BY Capacity DESC;

(iii) SELECT MAX(Charges) FROM carden;

(d) Give the output of the following SQL queries:

(i) SELECT COUNT(DISTINCT Make) FROM CARDEN;

(ii) SELECT MAX(Charges),MIN(Charges) FROM CARDEN;

(iii) SELECT COUNT(*),Make FROM CARDEN;

(iv) SELECT CarName FROM CARDEN WHERE Capacity=4;

Ans. (i) COUNT(DISTINCT Make)

4

(ii) MAX(Charges) MIN(Charges)

35 12

(iii) COUNT(*) Make

5 Suzuki

(iv) CarName

SX4

C Class

4. (a) Consider the following tables EMPLOYEE and SALGRADE and answer (b) and (c) parts of this question.

Table: EMPLOYEE

ECODE	NAME	DESIG	SGRADE	DOJ	DOB
101	Abdul Ahmad	EXECUTIVE	S03	23-Mar-2003	13-Jan-1980
102	Ravi Chander	HEAD-IT	S02	12-Feb-2010	22-Jul-1987
103	John Ken	RECEPTIONIST	S03	24-Jun-2009	24-Feb-1983
105	Nazar Ameen	GM	S02	11-Aug-2006	03-Mar-1984
108	Priyam Sen	CEO	S01	29-Dec-2004	19-Jan-1982

Table: SALGRADE

SGRADE	SALARY	HRA
S01	50000	10000
S02	32000	14000
S03	24000	8000

(a) Write SQL commands for the following statements:

(i) To display the details of all employees in descending order of DOJ.

(ii) To display NAME and DESIG of those employees whose SALGRADE is either S02 or S03.

(iii) To display the content of the entire employees table, whose DOJ is between '09-Feb-2006' and '08-Aug-2009'.

(iv) To add a new row with the following content:

109, 'Harish Roy', 'HEAD IT', 'S03', '9-Sep-2007', '21-Apr-1983'

Ans. (i) SELECT * FROM employee ORDER BY DOJ DESC;

(ii) SELECT name, design FROM employee WHERE sgrade='S02' OR sgrade='S03';

(iii) SELECT * FROM employee WHERE DOJ BETWEEN '09-Feb-2006' AND '08-Aug-2009';

(iv) Insert into Employee values (109, 'Harish Roy', 'HEAD IT', 'S02', '9-Sep-2007', '21-Apr-1983');

(c) Give the output of the following SQL queries:

(i) SELECT COUNT(SGRADE),SGRADE FROM EMPLOYEE GROUP BY SGRADE;

(ii) SELECT MIN(DOB),MAX(DOJ) FROM EMPLOYEE;

(iii) SELECT SGRADE, SALARY+HRA FROM SALGRADE WHERE SGRADE ='S02';

Ans. (i)

COUNT (SGRADE)	SGRADE
2	S03
2	S02
1	S01

(ii)

MAX (DOB)	MIN (DOJ)
22-Jul-1987	23-Mar-2003

(iii)

SGRADE	SALARY+HRA
P003	440000

5. Consider the following tables STOCK and DEALERS and answer (B1) and (B2) parts of this question:

Table: STOCK

ItemNo	Item	Dcode	Qty	UnitPrice	StockDate
5005	Ball Pen 0.5	102	100	16	31-Mar-10
5003	Ball Pen 0.25	102	150	20	01-Jan-10
5002	Gel Pen Premium	101	125	14	14-Feb-10
5006	Gel Pen Classic	101	200	22	01-Jan-09
5001	Eraser Small	102	210	5	19-Mar-09
5004	Eraser Big	102	60	10	12-Dec-09
5009	Sharpener Classic	103	160	8	23-Jan-09

Table: DEALERS

Dcode	Dname
101	Reliable Stationers
103	Classic Plastics
102	Clear Deals

B1. Write SQL commands for the following statements:

(i) To display details of all items in the Stock table in ascending order of StockDate.

Ans. SELECT * FROM STOCK ORDER BY StockDate;

(ii) To display ItemNo and Item name of those items from Stock table whose UnitPrice is more than ₹ 10.

Ans. SELECT ItemNo, Item FROM STOCK WHERE UnitPrice>10;

(iii) To display the details of those items whose dealer code (Dcode) is 102 or Quantity in Stock (Qty) is more than 100 from the table Stock.

Ans. SELECT * FROM STOCK WHERE Dcode=102 OR Qty>100;

B2. Give the output of the following SQL queries:

(i) SELECT COUNT(DISTINCT Dcode) FROM Stock;

Ans.

Count(DISTINCT Dcode)
3

(ii) SELECT Qty*UnitPrice FROM Stock WHERE ItemNo=5006;

Ans.

Qty*UnitPrice
4400

(iii) SELECT MIN(StockDate) FROM Stock;

Ans.

MIN (StockDate)
01-Jan-09

UNSOLVED QUESTIONS

1. What is the difference between data and information?
2. What is database and database system? What are the elements of database system?
3. Why do we need a database?
4. What is database management system? Why do we need a DBMS?
5. Explain the difference between:
   (a) Database and file (b) Data and file
6. Describe the components of the database system.
7. What are the advantages and disadvantages of DBMS?
8. What is the main function of DBA?
9. What is data redundancy? How can it be controlled?
10. Write SQL queries to perform the following based on the table PRODUCT having fields as (prod_id, prod_name, quantity, unit_rate, price, city)
    (i) Display those records from table PRODUCT where prod_id is more than 100.
    (ii) List records from table PRODUCT where prod_name is 'Almirah'
    (iii) List all those records whose price is between 200 and 500.
    (iv) Display the product names whose price is less than the average of price.
    (v) Show the total number of records in the table PRODUCT.
11. Suppose that all the customers of a particular business live in states for which city name is unique. Given the following description for customer data:

    CUST-ID, CUST-NAME, STREET, CITY, STATE, PHONE)
    (a) List the most likely key for the primary key.
    (b) List all the candidate keys and alternate keys.
12. Define the following terms:
    (a) Relation (b) Domain
    (c) Tuple (d) Attribute
    (e) Degree (f) Cardinality

13. Define the following:
(a) primary key
(b) candidate key
(c) alternate key
(d) foreign key

14. Consider the following tables STORE and SUPPLIERS. Write SQL commands for the statements (a) to (d) and give outputs for SQL queries (e) to (g).

Table: STORE

ItemNo	Item	Scode	Qty	Rate	LastBuy
2005	Sharpener Classic	23	60	8	31-Jun-09
2003	Ball Pen 0.25	22	50	25	01-Feb-10
2002	Gel Pen Premium	21	150	12	24-Feb-10
2006	Gel Pen Classic	21	250	20	11-Mar-09
2001	Eraser Small	22	220	6	19-Jan-09
2004	Eraser Big	22	110	8	02-Dec-09
2009	Ball Pen 0.5	21	180	18	03-Nov-09

Table: SUPPLIERS

Scode	Sname
21	Premium Stationery
23	Soft Plastics
22	Tetra Supply

(a) To display details of all the items in the Store table in ascending order of LastBuy.
(b) To display Itemno and item name of those items from store table whose rate is more than 15 rupees.
(c) To display the details of those items whose supplier code is 22 or Quantity in store is more than 110 from the table Store.
(d) To display minimum rate of items for each Supplier individually as per Scode from the table Store.
(e) SELECT COUNT(DISTINCT Scode) FROM STORE;
(f) SELECT Rate*Qty FROM STORE WHERE Itemno=2004;
(g) SELECT MAX(LastBuy)FROM STORE;

15. Write SQL commands for (i) to (v) on the basis of relations given below:

BOOKS

book_id	Book_name	author_name	Publishers	Price	Type	qty
k0001	Let us C	Sanjay Mukharjee	EPB	450	Comp	15
p0001	Genuine	J. Mukhi	FIRST PUBL.	755	Fiction	24
m0001	Mastering C++	Kantkar	EPB	165	Comp	60
n0002	VC++ advance	P. Purohit	TDH	250	Comp	45
k0002	Programming with Python	Sanjeev	FIRST PUBL.	350	Fiction	30

ISSUED

Book_ID	Qty_Issued
L02	13
L04	5
L05	21

(i) To show the books of FIRST PUBL. Publishers written by P. Purohit.
(ii) To display cost of all the books published for FIRST PUBL.
(iii) Depreciate the price of all books of EPB publishers by 5%.
(iv) To show total cost of books of each type.
(v) To show the details of the costliest book.

16. Write SQL commands for (a) to (f) and write output for (g) on the basis of PRODUCTS relation given below:

PRODUCTS TABLE

PCODE	PNAME	COMPANY	PRICE	STOCK	MANUFACTURE	WARRANTY
P001	TV	BPL	10000	200	12-JAN-2018	3
P002	TV	SONY	12000	150	23-MAR-2017	4
P003	PC	LENOVO	39000	100	09-APR-2018	2
P004	PC	COMPAQ	38000	120	20-JUN-2019	2
P005	HANDYCAM	SONY	18000	250	23-MAR-2017	3

(a) To show details of all PCs with stock more than 110.

(b) To list the company which gives warranty of more than 2 years.

(c) To find stock value of the BPL company where stock value is the sum of the products of price and stock.

(d) To show number of products from each company.

(e) To count the number of PRODUCTS which shall be out of warranty on 20-NOV-2020.

(f) To show the PRODUCT name of the products which are within warranty as on date.

(g) Give the output of the following statements:

(i) Select COUNT(distinct company) from PRODUCT.

(ii) Select MAX(price)from PRODUCT where WARRANTY<=3

17. What are DDL and DML?

18. Differentiate between primary key and candidate key in a relation.

19. What do you understand by the terms Cardinality and Degree of a relation in relational database?

20. Differentiate between DDL and DML. Mention the two commands for each category.

21. Consider the given table and answer the questions.

Table: SchoolBus

Rtno	Area_Covered	Capacity	Noofstudents	Distance	Transporter	Charges
1	Vasant Kunj	100	120	10	Shivam travels	100000
2	Hauz Khas	80	80	10	Anand travels	85000
3	Pitampura	60	55	30	Anand travels	60000
4	Rohini	100	90	35	Anand travels	100000
5	Yamuna Vihar	50	60	20	Bhalla Co.	55000
6	Krishna Nagar	70	80	30	Yadav Co.	80000
7	Vasundhara	100	110	20	Yadav Co.	100000
8	Paschim Vihar	40	40	20	Speed travels	55000
9	Saket	120	120	10	Speed travels	100000
10	Janakpuri	100	100	20	Kisan Tours	95000

(a) To show all information of students where capacity is more than the no. of students in order of rtno.

(b) To show Area_covered for buses covering more than 20 km., but charges less than 80000.

(c) To show transporter-wise total no. of students travelling.

(d) To show Rtno, Area_covered and average cost per student for all routes where average cost per student is—Charges/Noofstudents.

(e) Add a new record with the following data:

(11, "Motibagh",35,32,10, "kisan tours", 35000)

(f) Give the output considering the original relation as given:

(i) select sum(distance) from schoolbus where transporter= "Yadav travels";

(ii) select min(noofstudents) from schoolbus;

(iii) select avg(charges) from schoolbus where transporter= "Anand travels";

(g) Select distinct transporter from schoolbus;