8 Relational Database and SQL

8.1 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**. For example, all files belonging to an organization will be treated as the database of that organization. A database, therefore, is considered as a repository of stored data.

We will now discuss some components like files, tables, records, fields, etc., that are an important part of a database.

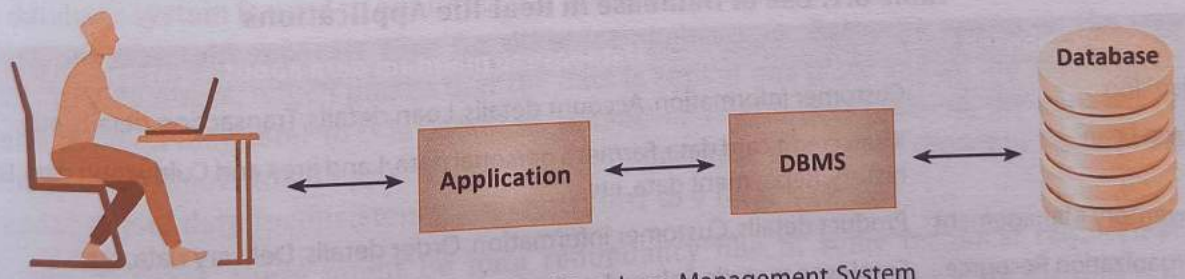


Fig. 8.1: Database and Database Management System

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. 8.2.

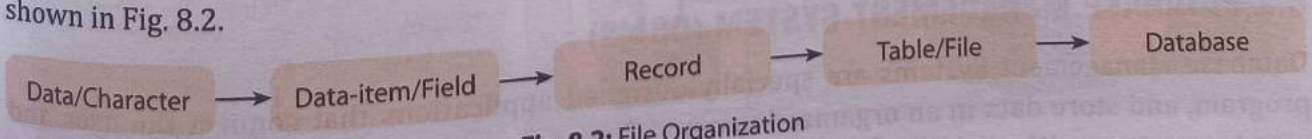


Fig. 8.2: File Organization

- **Data/character** is the smallest unit of file organization which represents itself in the form of a bit that may be either 0 or 1. Eight bits make a byte which represents a character in a computer.
- 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.

  For example, roll number, name, age and marks are the fields in a student's record.
- A collection of fields is termed as a **Record**. For example, a Student record consists of the fields Roll No, Name, Age and Marks as shown in Fig. 8.3.

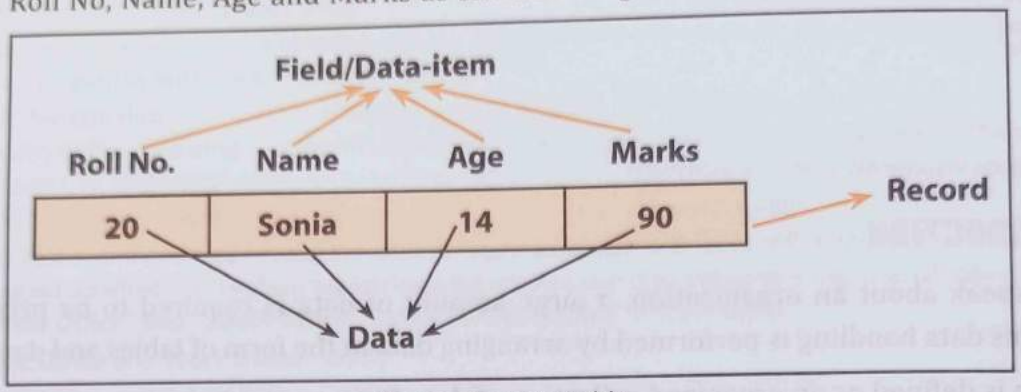


Fig. 8.3: Student Table

- 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 the 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. 8.3).
- **Database** is, therefore, a place where related information is stored and various operations can be performed on it. It is the highest unit of file organization.

Table 8.1: Use of Database in Real-life Applications

Application	Database to maintain data about
1. Banking	Customer information, Account details, Loan details, Transaction details, etc.
2. Crop Loan	Kisan credit card data, Farmer's personal data, Land area and Cultivation data, Loan history, Repayment data, etc.
3. Inventory Management	Product details, Customer information, Order details, Delivery data, etc.
4. Organization Resource Management	Employee records, Salary details, Department information, Branch locations, etc.
5. Online Shopping	Item description, User login details, User preference details, etc.

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

8.2 DATABASE MANAGEMENT SYSTEM (DBMS)

Database Management Systems are specially-designed applications that connect the user and program, and 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. It maintains the various pieces of information in an integrated and summarized form instead of keeping them in separate independent files.

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

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

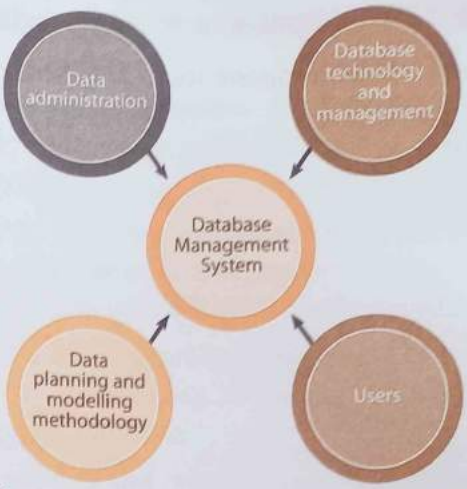


Fig. 8.4: Role of Database Management System

A DBMS gives us tools to:

- Store data in a structured way.
- Query the database (*i.e.*, ask questions about the data).
- Sort and manipulate the data in the database.
- Validate the data entered and check for inconsistencies.
- Produce flexible reports, both on screen and on paper, that make it easy to comprehend the information stored in the database.

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

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

8.2.1 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. 8.5. 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 same data for different files but this is not recommended.

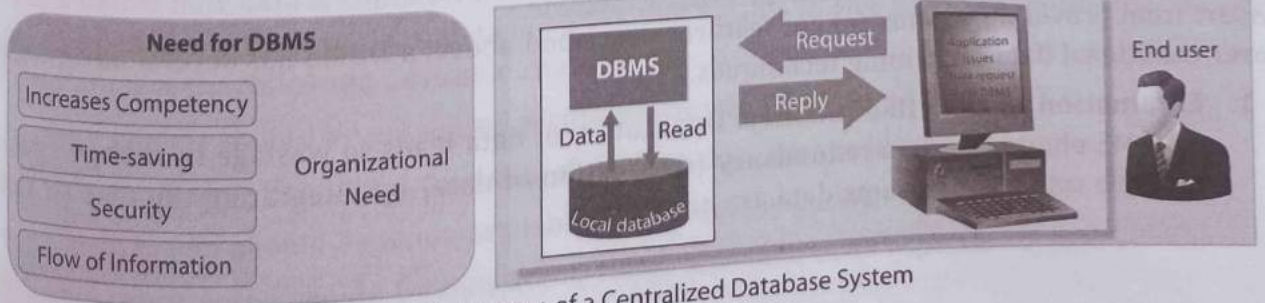


Fig. 8.5: Purpose of a Centralized Database System

8.2.2 Components of a Database System

The various components of a database system are described in Fig. 8.6 below:

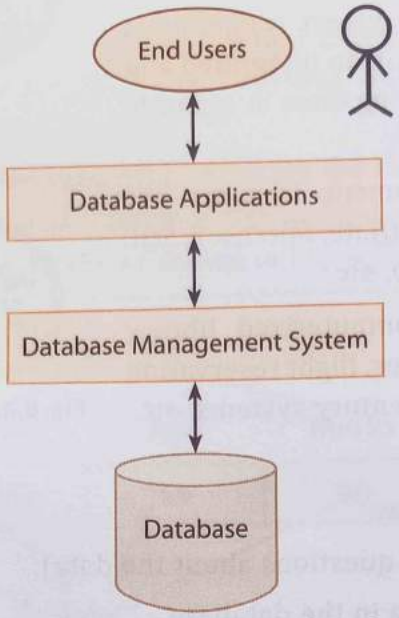


Fig. 8.6: Components of a Database System

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 the following 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:** It helps in 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 users and retrieves necessary information from the database.
2. **Database Application:** Database application may be Personal, Departmental, Enterprise and Internal. It may be general-purpose or customized as per the needs of a 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:** It is a collection of logically related data.

8.2.3 Advantages of a DBMS

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

1. **Elimination of Data Redundancy:** Duplication of data leads to wastage in 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:** A 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:** By using a DBMS, not only can existing applications share data in the database, but new applications can also 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 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 work on this single source of data can result in cost savings. 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 failures.
10. **Meeting Enterprise Requirements than Individual Requirements:** Since many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces.

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

8.3 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 the appropriate data types, relationships and constraints required to be applied on the data.

Data models are used to implement abstraction in a DBMS. They are a communication tool. Data models define how data is connected and how it is processed and stored inside the system. They organize data for various users. A data model should be able to give 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. A data model should be made available and shared by various applications.
4. Data represented should be consistent, stable and valid in all aspects.

8.3.1 Types of Data Models

Data models are categorized into three different categories:

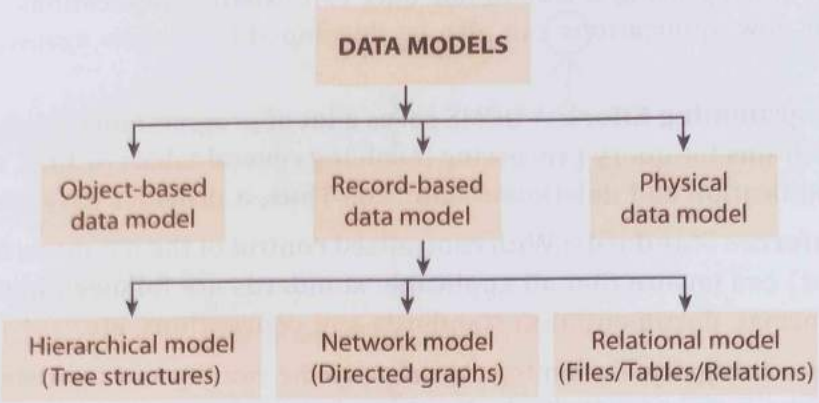


Fig. 8.7: Classification of Data Models

8.3.1.1 Hierarchical Data Model

In a hierarchical model, records are organized as trees rather than graphs, *i.e.*, a hierarchical model represents a hierarchy of parent and child data segments. It is represented by an upside-down "tree". Hierarchical model depicts one-to-many (1:M) relationships between a parent and child segment. Each parent can have many children, but vice versa does not hold true, *i.e.*, each child has only one parent.

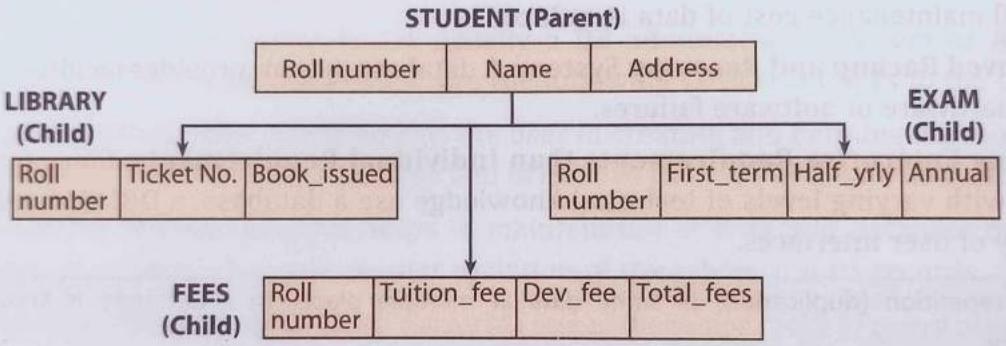


Fig. 8.8: Hierarchical Model

From Fig. 8.8, we can see that the student database contains various files related to a student. In this kind of arrangement, a file higher in the hierarchy is known as parent of the files contained inside it. Here, Student is a parent of the Library, Fees and Exam files respectively. Thus, this model represents a one-to-many relationship between a parent and its children in the form of a hierarchical (upside-down) tree. The **advantages** of this model are:

1. It is simple in operation and in concept.
2. It promotes data sharing.
3. Parent-child relationship promotes data integrity.
4. It is efficient and can be naturally mapped for 1:M (one-to-many) relationships.

The **limitations** of this model are as follows:

1. It is suitable for hierarchical relationships only; otherwise, it is inflexible for non-hierarchical relationship mapping.
2. Implementation is complex and difficult since it is done using pointers from a parent to its children which require extra memory space.

3. Dependency on parent node, which is the root node, and its deletion can cause deletion of all child nodes and, in turn, the entire model.
4. Changes in the structure require changes in all the applications.

IBM's IMS (Information Management System) and system2000 are examples of hierarchical database management system.

8.3.1.2 Network Data Model

The network model is a database model conceived as a flexible way of representing objects and their relationships. It consists of a collection of records connected to one another through **links**. The network model replaces the hierarchical model with a graph, thus allowing multiple connections among the nodes. The major point of difference between the two models is that unlike the hierarchical model, the network model provides the capability to handle many-to-many (M:M) relationships. Thus, a child can have more than one parent.

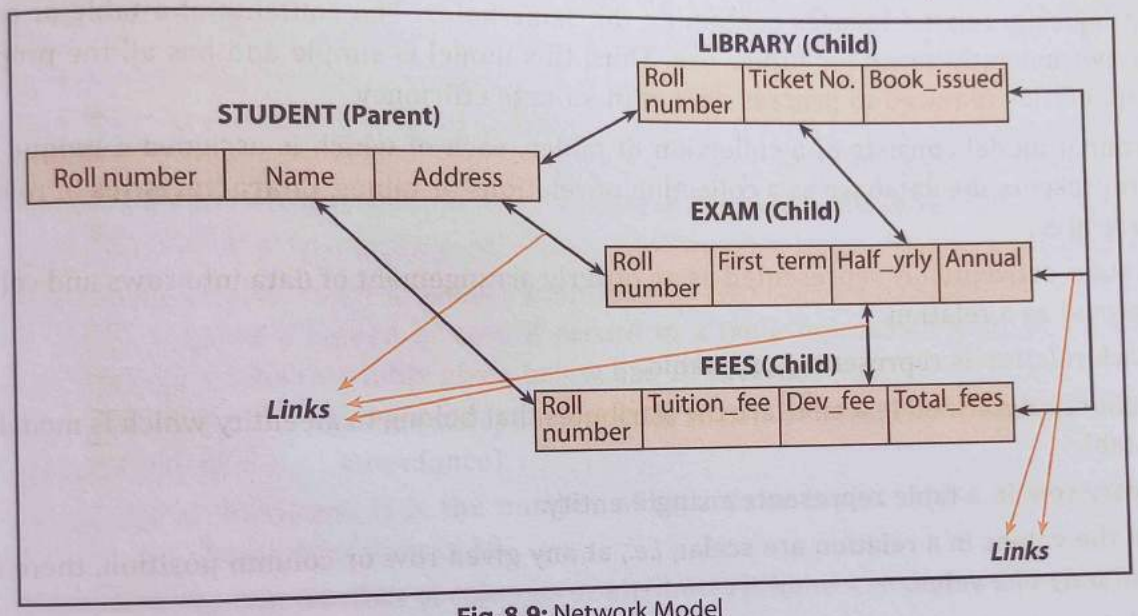


Fig. 8.9: Network Model

From Fig. 8.9, it is clear that this model represents many-to-many relationships. It can be observed that the file EXAM is associated with three tables, viz. LIBRARY, FEES and STUDENT. Thus, the figure shows that a child can have more than one parent. This model is an improved version of hierarchical model having records along with pointers. These pointers establish many-to-many relationships and are termed as **Links**. Each record has its respective pointer or link with which it is associated.

The **advantages** of network model are:

1. It can handle more relationship types, *i.e.*, M:M (many-to-many) multi-parent relationships.
2. Data access is easy and more flexible.
3. It includes DDL (Data Definition Language) as well as DML (Data Manipulation Language).
4. It promotes better data integrity.

The **limitations** of this model are:

1. The system is quite complicated in structure since there are several number of links which also limits efficiency.
2. Structural changes require changes to be made in all application programs.

3. A high-level language interface is required to interact with the database.
4. Extra memory is consumed for holding links (pointers).

UNIVAC's DMS 1100 and IDMS (Integrated Database Management System) are examples of network database management system.

8.3.1.3 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 can 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.
6. All operations are performed on the entire relation and the result is an entire relation.

Advantages of a relational model are as follows:

1. A relational model provides structural independence by using independent tables.
2. Changes made in the table structure do not affect the data access or other application programs.
3. It is represented in the form of tables; so, it is simple and easier to understand.
4. Tabular view also provides easier 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.

The **limitations** of a relational model are:

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

8.4 RELATIONAL DATABASE

A **relational database** is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a record with a unique ID called the key. The columns of the table hold attributes of the data and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

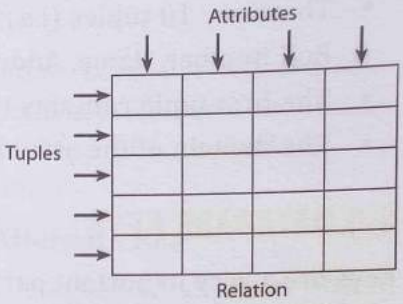


Fig. 8.10: Tuples and Attributes

Often, data in a relational database is organized into tables.

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. **Attribute:** 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 (row); 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 (Fig. 8.11) 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 given below has 10 records.

   Each tuple (row) in a relation (table) corresponds to data of a real world entity (for example, Student, Employee and Attendance)
4. **Cardinality of Relation:** It is the number of records or tuples in the relation. Thus, the cardinality of Student relation is 10.
5. **Degree of Relation:** Number of columns or attributes is known as 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.

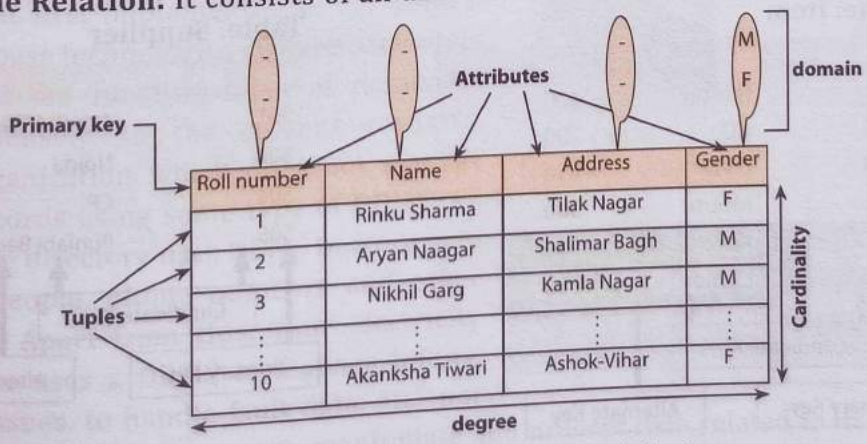


Fig. 8.11: Relational Model (Student Relation)

Therefore, with reference to Fig. 8.11, in the given 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).

8.5 DATABASE KEYS

Keys are a very important part of relational database. They allow 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 among two or more tables. They also ensure that each record 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 is 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 can be declared as a primary key in the relation Student. In the table Item given below, Item_id is the primary key while Supp_id (supplier id) is the primary key in the table Supplier.

Table: Item

Item_id	Item_name	Qty
I101	Printer	400
I102	CD	200
I104	DVD	150
I105	Mouse	300
I103	Keyboard	180
I109	Cable	500

Candidate Keys

Primary Key

Alternate Key

Table: Supplier

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

Candidate Keys

Primary Key

Alternate Key

Fig. 8.12: Keys in a Database (Tables)

2. **Candidate Key:** A candidate key refers to all the attributes in a relation that are candidates or are capable of becoming a primary key. We mark it virtually.

In the given 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 given 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, a primary key in some other table having relationship with the current or original table.

Foreign Key

Table: Employee

Item_id	Emp_name	Desig_code
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.

This was all about the database concepts. Now, we will be moving on to their implementation using SQL.

8.6 INTRODUCTION TO SQL

Online Telephone Directory

Electricity Billing System

Database Management System (DBMS) is not a new concept and was first implemented in the 1960's. With time, database technologies evolved a lot while usage and expected functionalities of databases increased immensely. In the current scenario, there is no organization which does not manage its data and records using some type of DBMS. An online telephone directory uses 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., not to forget Facebook—it needs to store, manipulate and present data related to its members, their friends, member activities, messages, advertisements and a lot more.

All these real-life applications require a DBMS to manipulate and handle this enormous data. A DBMS requires some language to handle and manipulate its data, which is known as Structured Query Language (SQL). The following topic deals exclusively with relational databases, their tables and retrieving data using SQL.

8.7 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 are also used to create interface between a user and database. By using SQL commands, one can search for any data in the database and perform other functions like creating tables, adding records, modifying data, removing rows, dropping tables, etc.

Fig. 8.13: Structured Query Language

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 freely and easily downloaded from the site www.mysql.org. It is fully secured, reliable and fast, and possesses far better functionalities than many other commercial RDBMSs available in the market. Originally, MySQL was developed and supported by a Sweden-based company, MySQL AB, which was bought by Sun Microsystems. In 2010, Oracle acquired Microsystems. The chief inventor of MySQL was Michael 'Monty' Widenius.

8.8 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 and updated or deleted from a database.

It has the following salient features and strong processing capabilities:

- It can retrieve data from a database through Query processing.
- It can insert records in a database.
- It can update records in a database.
- It can create new databases and modify the existing ones.
- It can create new tables in a database.
- It can create views in a database.
- It allows modifying the security settings of the system.

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

8.9 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 to work upon it.
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.

8.10 CLASSIFICATION OF SQL STATEMENTS

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

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

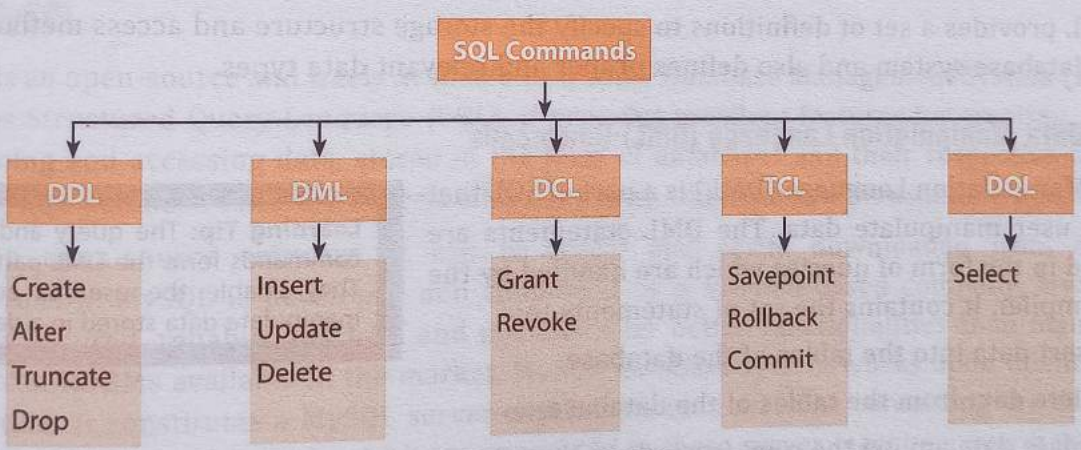


Fig. 8.14: Classification of SQL Statements

DCL and TCL are beyond the scope of this book. So, we shall be discussing only DDL, DQL and DML commands in detail.

8.10.1 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 necessary statements for creating, manipulating, altering and deleting the table.

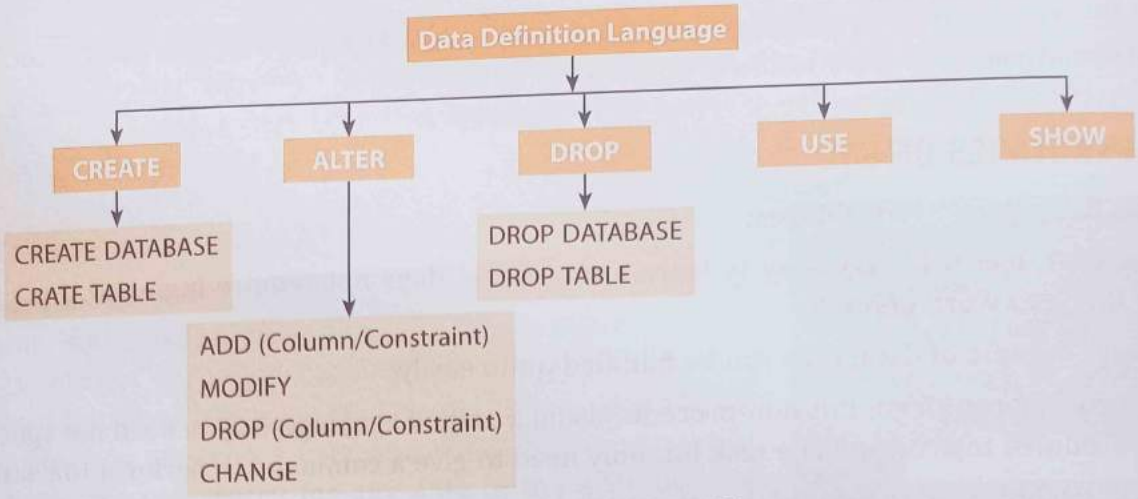


Fig. 8.15: DDL Commands in SQL

Examples of DDL commands in SQL are:

- **CREATE DATABASE:** creates a new database.
- **USE command:** to select and open an already existing database.
- **SHOW command:** to display all the tables in an existing database.
- **CREATE TABLE:** creates a new table in an existing database.
- **ALTER TABLE:** modifies the structure of a table.
- **DROP TABLE:** deletes a table.

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

The 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.

8.10.2 Data Manipulation Language (DML) Commands

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

1. Insert data into the tables of the database.
2. Delete data from the tables of the database.
3. Update data among the rows/records in the tables of the database.

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

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

1. **INSERT INTO statement:** To insert new data (record) into a table.
2. **UPDATE statement:** To modify or change the data (tuple) in a table (not modifying the data type of column).
3. **DELETE:** To delete data (tuple) from a table (not deleting a column).

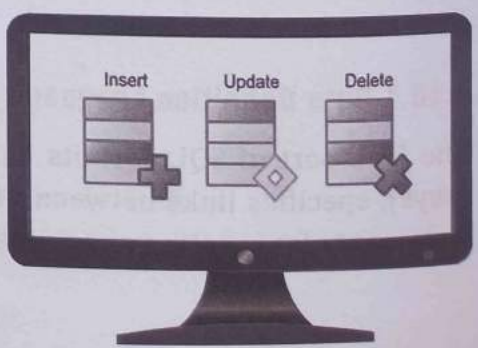


Fig. 8.16: DML Commands in SQL

Table 8.2: 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. The examples of DDL commands are Create, Alter, Drop, etc.	3. The examples of DML commands are Insert into, Update, Delete, etc.
4. DDL is not further classified.	4. DML commands are further classified into two types: (a) Procedural DMLs (b) Non-Procedural DMLs

8.10.3 Data Query Language (DQL) – SELECT Command

One of the most important tasks when working with SQL is to generate queries and retrieve data. A query is a command given to get a desired result from the database table. The SELECT command is used to query or retrieve data from a table in the database. It is used to retrieve a subset of records from one or more tables. The SELECT command can be used in various forms:

Syntax of **SELECT** command:

SELECT <column-list>FROM<table-name>;

column-list includes one or more columns from which data is retrieved.

We will be discussing the **SELECT command** in detail in the successive sections.

Let us start implementing SQL using MySQL as the platform.

8.11 MySQL

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 freely and easily downloaded from the site *www.mysql.org*. MySQL is developed and supported by a Sweden-based company, MySQL AB. It is fully secured, reliable, and fast, and possesses far better functionalities than many other commercial RDBMs available in the market. MySQL database system works upon 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 provides 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 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/constraints in order to restrict the user input and processing.
6. **Authorization:** MySQL provides DDL commands to check for user authentication and authorization by restricting access to relations and views.
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 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.

8.11.1 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

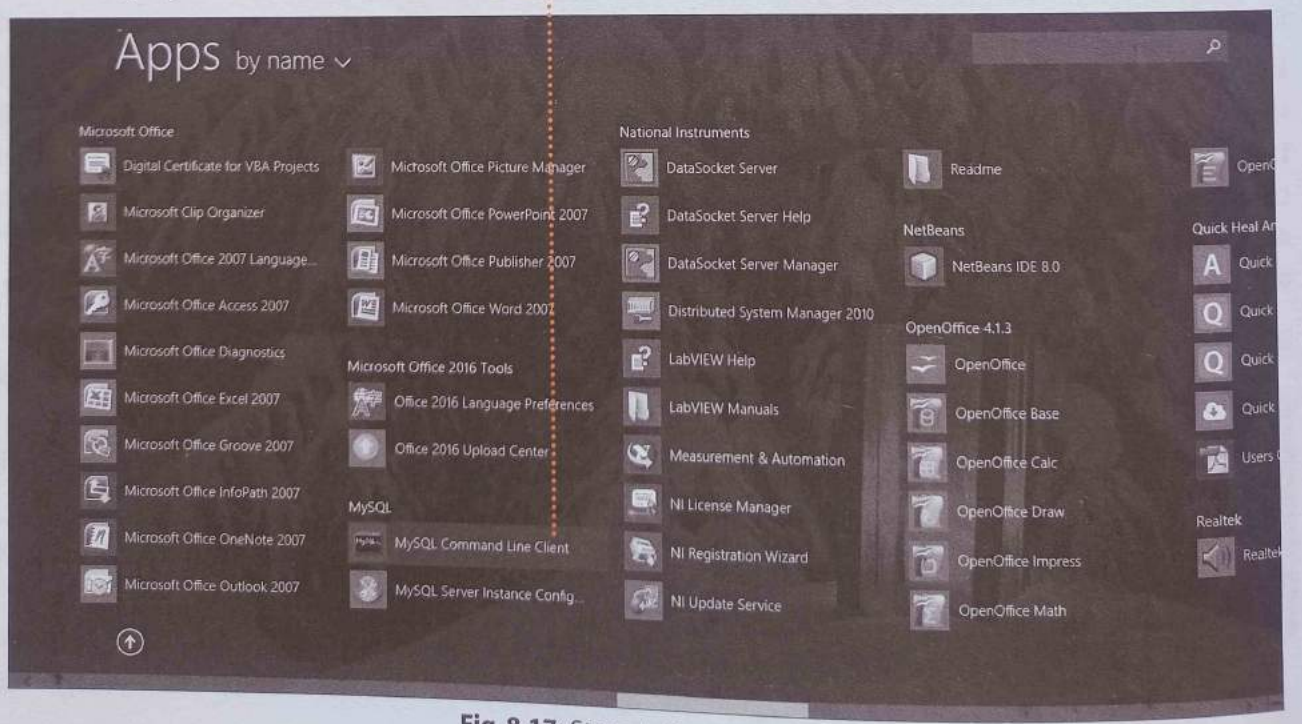


Fig. 8.17: Steps to Start MySQL

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

After entering the password, the MySQL prompt appears, where you start typing the SQL commands, as shown in Fig. 8.17. In order to come out of MySQL application, you can type quit in front of the mysql> command prompt, as shown in Fig. 8.18.

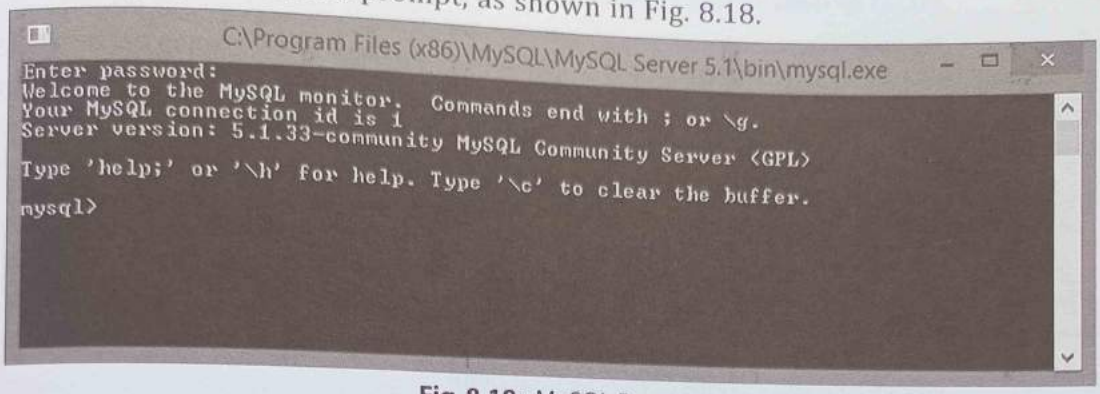


Fig. 8.18: MySQL Prompt

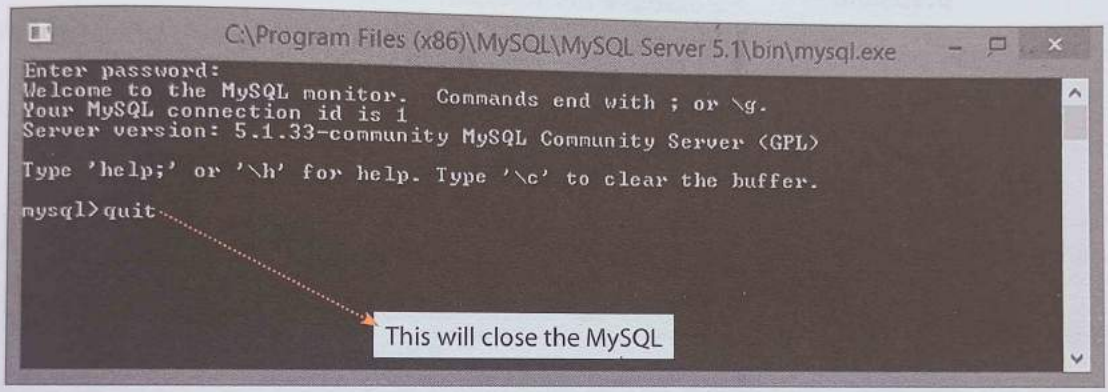


Fig. 8.19: Closing MySQL

Learning Tips:

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

8.12 SQL DATA TYPES

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.

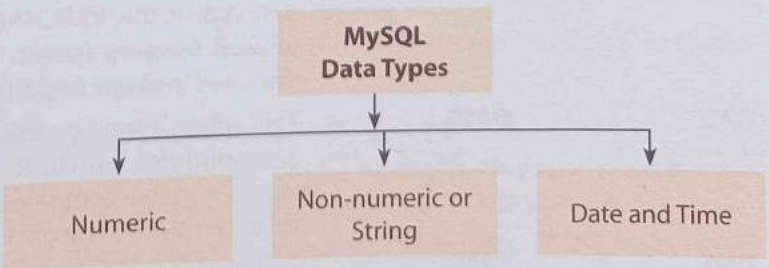


Fig. 8.20: MySQL Data types

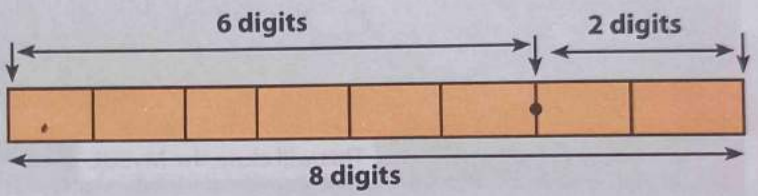
Data type	Syntax	Description
INTEGER (Numeric)	**INTEGER or integer**	It stores/represents positive whole numbers up to 11 digits and negative whole numbers up to 10 digits. The range of integer is from –2,147,483,648 to 2,147,483,647.
FLOAT	**FLOAT or float**	Holds numbers with decimal points. Each FLOAT value occupies 4 bytes.
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)** **decimal point**	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 6 digits before the decimal and 2 digits after the decimal. Decimal holds up to 19 significant digits. A negative number uses one place for its sign (–). 
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 store is fixed. The value for CHAR data type has to be enclosed in single or double quotation marks.
CHARACTER (variable length)	**VARCHAR(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. The value for VARCHAR data type has to be enclosed in single or double quotation marks.
DATE	**DATE**	This data type is used to store a date in 'yyyy/mm/dd' or '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 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, a time of day can be taken as 12:30:45 where 12 means hours, 30 means minutes and 45 refers to seconds.

Table 8.3: 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 number up to 65,535.
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 to that column.	4. VARCHAR(x) will take only the required storage for the actual number of characters entered to that column.
5. If a value entered is shorter than its length x, then blanks are added.	5. No blanks are added if the length is shorter than the maximum length, x.
6. CHAR data type takes memory space of 1 byte per character.	6. VARCHAR takes up memory space of 1 byte per character, +2 bytes to hold variable length information.
7. Search operation is faster with CHAR data type column.	7. Search operation works slower in VARCHAR data type column as compared to CHAR type.
8. For example, name char(10); name="anu"; name field occupies 10 bytes, with the first three bytes with values and the rest with blank data.	8. For example, name varchar(10); name="anu"; then name occupies only 3+2=5 bytes, the 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.

8.13 CONSTRAINTS IN SQL

Constraints are some set of rules created to apply data validations. They ensure accuracy, correctness and reliability of data. However, it is not mandatory to define constraints for each attribute of a table. SQL provides the following constraints:

Table 8.4: Commonly-used SQL Constraints

Constraint	Description
Primary Key	The field or column or attribute which uniquely identifies each row/record in a table.
NOT NULL	When this constraint is applied to a field/column, it ensures that the column cannot have NULL values. NULL means missing/unknown/not applicable value.
UNIQUE	It contains unique records for the column except for the primary key. In other words, it ensures that all the values in a column are distinct/unique.
DEFAULT	The default constraint allows to assign a default value if the value is not provided or given.
FOREIGN KEY	The column which refers to value of an attribute defined as the primary key in another table.

8.14 SQL COMMANDS

SQL provides a predefined set of commands that help us to work with relational databases. Before discussing these commands, we must be familiar with the conventions and basic terminologies related to SQL. Throughout this chapter, the words keyword, clause and statement have been used.

A **keyword** refers to an individual SQL element that has a special meaning in SQL. For example, SELECT and FROM are keywords. A **clause** is a distinct logical part of an SQL statement. Clauses begin with a keyword for which they are named and consist of arguments as well. For example, SELECT empno, ename FROM employee WHERE salary>45000, are clauses in SQL. Here, arguments complete or modify the meaning of a clause, which is salary in the given example.

A **statement** or **command** is a combination of two or more clauses. Statements are basically the instructions given to SQL database for executing any task. For example, SELECT * FROM employee; is an SQL statement. An important point to remember here is that all the statements in SQL terminate with a semi-colon (;). Also, SQL is not case sensitive; therefore, we can type commands in either upper case or lower case.

Let us now learn how a database and tables in a 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 sample database School and then create a table Student in it.

Database: School

Tables_in_School
Student

Table: Student

Rollno	Name	Gender	Marks	DOB
1	Raj Kumar	M	93	2000-11-17
2	Deep Singh	M	98	1996-08-22
3	Ankit Sharma	M	76	2000-02-02
4	Radhika Gupta	F	78	1999-12-03
5	Payal Goel	F	82	1998-04-21
6	Diksha Sharma	F	80	1999-12-17
7	Gurpreet Kaur	F	65	2000-01-04
8	Akshay Dureja	M	90	1997-05-05
9	Shreya Anand	F	70	1999-10-08
10	Prateek Mittal	M	75	2000-12-25

To get started on 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 by default on the server:

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

Let us discuss the important DDL (Data Definition Language) commands.

8.14.1 DDL Commands

The data stored is organized in the form of well-defined schemas or relations or what are most commonly referred to as tables.

SQL allows us to write statements for defining, modifying and deleting relation schemas. These are part of Data Definition Language (DDL).

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,

```
mysql> 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 DATABASE** 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 comma. Upper case and lower case letters make no difference in column names.

5. Viewing a Table

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

```
mysql> SHOW TABLES;
+----------------------+
| Tables_in_school     |
+----------------------+
| Student              |
+----------------------+
2 rows in set (0.01 sec)
```

6. Viewing a Table Structure

To view a table structure, **DESCRIBE** or **DESC** command is used. It shows the structure of the table along with the name of the columns, data type of the columns and constraints applied on the columns.

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

For example,

```
mysql> DESCRIBE Student;
```

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

5 rows in set (0.02 sec)

7. 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><data type> [size]);

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

```
mysql> ALTER TABLE Student ADD(Mobile_no integer);
```

Thus, the above statement shall add a new column Mobile_no into 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) **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,

```
mysql> 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	2000-11-17	NULL	DELHI
2	Deep Singh	M	98	1996-08-22	NULL	DELHI
3	Ankit Sharma	M	76	2000-02-02	NULL	DELHI
4	Radhika Gupta	F	78	1999-12-03	NULL	DELHI
5	Payal Goel	F	82	1998-04-21	NULL	DELHI
6	Diksha Sharma	F	80	1999-12-17	NULL	DELHI
7	Gurpreet Kaur	F	65	2000-01-04	NULL	DELHI
8	Akshay Dureja	M	90	1997-05-05	NULL	DELHI
9	Shreya Anand	F	70	1999-10-08	NULL	DELHI
10	Prateek Mittal	M	75	2000-12-25	NULL	DELHI

(c) **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] <data type1>);

For example,

```
mysql> 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.

(d) **Renaming a column**

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

Syntax for renaming an existing column:

ALTER TABLE <table_name>

CHANGE[COLUMN]<old-column-name><new-column-name>column_definition;

For example,

```
mysql> ALTER TABLE Student CHANGE City State varchar(10);
```

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	2000-11-17	NULL	DELHI
2	Deep Singh	M	98	1996-08-22	NULL	DELHI
3	Ankit Sharma	M	76	2000-02-02	NULL	DELHI
4	Radhika Gupta	F	78	1999-12-03	NULL	DELHI
5	Payal Goel	F	82	1998-04-21	NULL	DELHI
6	Diksha Sharma	F	80	1999-12-17	NULL	DELHI
7	Gurpreet Kaur	F	65	2000-01-04	NULL	DELHI
8	Akshay Dureja	M	90	1997-05-05	NULL	DELHI
9	Shreya Anand	F	70	1999-10-08	NULL	DELHI
10	Prateek Mittal	M	75	2000-12-25	NULL	DELHI

(e) **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,

```
mysql> 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	2000-11-17	NULL
2	Deep Singh	M	98	1996-08-22	NULL
3	Ankit Sharma	M	76	2000-02-02	NULL
4	Radhika Gupta	F	78	1999-12-03	NULL
5	Payal Goel	F	82	1998-04-21	NULL
6	Diksha Sharma	F	80	1999-12-17	NULL
7	Gurpreet Kaur	F	65	2000-01-04	NULL
8	Akshay Dureja	M	90	1997-05-05	NULL
9	Shreya Anand	F	70	1999-10-08	NULL
10	Prateek Mittal	M	75	2000-12-25	NULL

(f) **Adding and Deleting Primary Key constraints**

We can add or delete a Primary Key constraint even if the table has already been created using ALTER TABLE command.

For example, add a Primary Key constraint on the column Emp_ID in the table Employee.

```
mysql> ALTER TABLE EMPLOYEE ADD PRIMARY KEY(Emp_ID);
```

Note: Make sure while using ALTER TABLE command to add a primary key that the primary key column(s) must be declared with NOT NULL constraint (when the table was first created).

- To delete a Primary Key constraint from the table Employee:

```
mysql> ALTER TABLE EMPLOYEE DROP PRIMARY KEY;
```

8. 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. If you drop a table, all the rows in the table are deleted along with its structure. Once a table is dropped, we cannot get it back and all other references to the table become invalid. This command completely destroys 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.

8.14.2 DML Commands

When we create a table, only its structure is created but the table has no data. To populate records in the table, INSERT statement is used. Also, table records can be deleted or updated using DELETE and UPDATE statements. These SQL statements are part of Data Manipulation Language (DML).

Data Manipulation using a database means either insertion of new data, removal of existing data or modification of existing data in the database.

1. 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 different forms:

(a) **Inserting 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:

INSERT INTO <table_name> VALUES (*value1, value2, value3...*);

For example,

```
mysql> INSERT INTO Student VALUES (1,"Raj Kumar", 'M', 93,
'2000-11-17');
```

While inserting a row, if we are adding value for all the columns of the table, we need not specify the column(s) name in the SQL query. But we 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 or text and date datatype are always enclosed in double or single quotation marks.
- Column values for the date data type of a column are provided within single quotes in 'yyyy-mm-dd' or "yyyy/mm/dd" format.

- 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 pressing up and down arrow keys followed by Enter key.

(b) **Inserting data by specifying all the column names and associated values 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,valueN,...);

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

For example,mysql> `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 specific columns of a table:**

Syntax for SQL INSERT is:

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

VALUES [(value1, value2,.... valueN)];

For example, to insert a record into the student table for the columns Rollno, Name and Marks only, the SQL insert query is:

mysql> `INSERT INTO Student (Rollno, Name, Marks) VALUES (4,"Radhika Gupta",78);`

The above statement shall insert the values for specific columns—Rollno, Name and Marks respectively. Also, the columns Gender and DOB shall hold the values as NULL.

(d) **Inserting NULL values into a table:** If a column in a row has no value or missing value, then the column is said to be null or holding NULL value. Null value can be given to any column other than being assigned as primary key or Not Null constraint. It is advisable to use Null when the actual value is not defined or unavailable. 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, mysql> `INSERT INTO Student(Rollno, Name, Gender, Marks, DOB)`
`VALUES(12, 'Swati Mehra', 'F', NULL, NULL);`

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

CTM: Null means unavailable or undefined value. Any arithmetic expression containing a NULL always evaluates to null.

2. 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 <condition>;

For example,
```
mysql> 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,
```
mysql> 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,
```
mysql> UPDATE Student
       SET Marks = NULL
       WHERE Rollno = 9;
```

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

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

For example,
```
mysql> 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.

(d) **Updating all rows using an expression or formula**

For example, mysql> `UPDATE Student`
`SET Marks = Marks + 10;`

The above statement shall increase the value of Marks of all the students by 10.

3. 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>;

here <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 it 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, mysql> `DELETE FROM Student WHERE Rollno = 10;`

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

- Deleting all rows from the table:

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

mysql> `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:

mysql> `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.

8.15 SQL QUERY PROCESSING

After creating the database, the table is created and the data is stored in it. Now, it is time to perform query processing on the 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.

8.15.1 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.

1. Projection

This capability of SQL returns all rows from a relation with selective attributes.

Syntax:

SELECT <column-name1> [, <column-name2>...]
FROM <table-name>;

For example,

```
mysql> SELECT Name, Gender FROM Student;
```

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

10 rows in a set (0.01 sec)

The above command displays only Name and Gender attributes from the student table.

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

To display all columns along with the respective rows, we use the command:

```
mysql> SELECT * FROM Student;
```

Resultant table: Student

Rollno	Name	Gender	Marks	DOB	Mobile_no
1	Raj Kumar	M	93	2000-11-17	NULL
2	Deep Singh	M	98	1996-08-22	NULL
3	Ankit Sharma	M	76	2000-02-02	NULL
4	Radhika Gupta	F	78	1999-12-03	NULL
5	Payal Goel	F	82	1998-04-21	NULL
6	Diksha Sharma	F	80	1999-12-17	NULL
7	Gurpreet Kaur	F	65	2000-01-04	NULL
8	Akshay Dureja	M	90	1997-05-05	NULL
9	Shreya Anand	F	70	1999-10-08	NULL
10	Prateek Mittal	M	75	2000-12-25	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.
- 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.

2. Selection: Selecting Specific Rows—WHERE Clause

This capability of SQL returns some of the rows and all the columns 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.

SELECT <what_to_select>

FROM <which_table>

WHERE <conditions_to_satisfy>;

For example,

```
mysql> SELECT * FROM Student WHERE Gender = 'M';
```

Resultant table: Student

Rollno	Name	Gender	Marks	DOB	Mobile_no
1	Raj Kumar	M	93	2000-11-17	NULL
2	Deep Singh	M	98	1996-08-22	NULL
3	Ankit Sharma	M	76	2000-02-02	NULL
8	Akshay Dureja	M	90	1997-05-05	NULL
10	Prateek Mittal	M	75	2000-12-25	NULL

5 rows in a set (0.01 sec)

The above command will display all attributes but specific tuples (rows) which satisfy the condition in the Student table.

Using WHERE clause

```
mysql> 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	2000-11-17	NULL
2	Deep Singh	M	98	1996-08-22	NULL
3	Ankit Sharma	M	76	2000-02-02	NULL
4	Radhika Gupta	F	78	1999-12-03	NULL
5	Payal Goel	F	82	1998-04-21	NULL
6	Diksha Sharma	F	80	1999-12-17	NULL
7	Gurpreet Kaur	F	65	2000-01-04	NULL
8	Akshay Dureja	M	90	1997-05-05	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).

CTM: SQL is case-insensitive, which means keywords like SELECT and select have same meaning in SQL statements.

3. Re-ordering 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 the display purpose and no actual (physical) rearrangement of the columns is done.

For example,

```
mysql> SELECT Name, Rollno, DOB, Marks FROM Student;
```

After executing the above statement, the column shall be displayed in the changed order as 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	2000-11-17	93
Deep Singh	2	1996-08-22	98
Ankit Sharma	3	2000-02-02	76
Radhika Gupta	4	1999-12-03	78
Payal Goel	5	1998-04-21	82
Diksha Sharma	6	1999-12-17	80
Gurpreet Kaur	7	2000-01-04	65
Akshay Dureja	8	1997-05-05	90
Shreya Anand	9	1999-10-08	70
Prateek Mittal	10	2000-12-25	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	2000-11-17	9586774748	Science
2	Deep Singh	M	98	1996-08-22	8988886577	Commerce
3	Ankit Sharma	M	76	2000-02-02	NULL	Science
4	Radhika Gupta	F	78	1999-12-03	9818675444	Humanities
5	Payal Goel	F	82	1998-04-21	9845639990	Vocational
6	Diksha Sharma	F	80	1999-12-17	9897666650	Humanities
7	Gurpreet Kaur	F	65	2000-01-04	7560567890	Science
8	Akshay Dureja	M	90	1997-05-05	9560567890	Commerce
9	Shreya Anand	F	70	1999-10-08	NULL	Vocational
10	Prateek Mittal	M	75	2000-12-25	9999967543	Science

10 rows in a set (0.02 sec)

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

```
mysql> 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 the field Stream from student table:

For example,

```
mysql> SELECT DISTINCT Stream FROM
Student;
```

Resultant table: Student

Stream
Science
Commerce
Science
Humanities
Vocational
Humanities
Science
Commerce
Vocational
Science

Science displayed 4 times

(Displays all rows with duplicate values as well)

10 rows in a set (0.02 sec)

Resultant table: Student

Stream
Science
Commerce
Humanities
Vocational

Science displayed once only

(Duplicate values are removed)

4 rows in a set (0.02 sec)

8.15.2 SQL Operators

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

(a) Arithmetic Operators
(b) Relational Operators
(c) Logical Operators
(d) Special Operators

Table 8.5: SQL Operators and their functions

OPERATOR/FUNCTION	DESCRIPTION
ARITHMETIC OPERATORS	Used in arithmetic calculations and expressions. +, -, *, /, %, **
COMPARISON/ RELATIONAL OPERATORS	
=, >, <, >=, <=, <>	Used in Conditional expressions.
LOGICAL OPERATORS	
AND/OR/NOT	Used in Conditional expressions.
SPECIAL OPERATORS	
BETWEEN/NOT BETWEEN	Checks whether an attribute value is within a range or not.
IS NULL/IS NOT NULL	Checks whether an attribute value is NULL or not.
LIKE/NOT LIKE	Checks whether an attribute matches a given string pattern or not.
IN/NOT IN	Checks whether an attribute value matches any value with a given list or not.
DISTINCT	Permits only unique values. Eliminates duplicate ones.
AGGREGATE FUNCTIONS	Used with SELECT to return mathematical results/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.

(b) 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.

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

The above statement returns the value 15 as the result.

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

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+----------------+------------+
| SIN(PI()/4)    | (4+1)*5    |
+----------------+------------+
| 0.707107       |    25      |
+----------------+------------+
```

```
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 shall return the value 2 as the output.

POINT TO REMEMBER

DUAL is the default table automatically created in MySQL. It has one row with a value X and one column 'dummy' defined as varchar2(!).

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 for 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	2000-11-17	9586774748	Science
3	Ankit Sharma	M	76	2000-02-02	8567490078	Science
4	Radhika Gupta	F	78	1999-12-03	9818675444	Humanities
5	Payal Goel	F	82	1998-04-21	9845639990	Vocational
6	Diksha Sharma	F	80	1999-12-17	9897666650	Humanities
7	Gurpreet Kaur	F	65	2000-01-04	7560567890	Science
9	Shreya Anand	F	70	1999-10-08	8876543988	Vocational
10	Prateek Mittal	M	75	2000-12-25	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 with the condition specified. If it is true, the corresponding row is selected for display, otherwise it is 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 the 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 operator. Out of these, AND and OR operators are termed as Conjunctive operators since these two operators 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, then only 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	2000-11-17	9586774748	Science
2	Deep Singh	M	98	2000-08-22	8988886577	Commerce
8	Akshay Dureja	M	90	1997-05-05	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, then only the result becomes 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 either of the conditions (usually two conditions) specified in the WHERE clause is true.

Condition 1	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)

(d) SQL Special Operators

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

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

SQL provides a BETWEEN operator that defines a range of values that the column value must fall within 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
7	Gurpreet Kaur	65
9	Shreya Anand	70
10	Prateek Mittal	75

5 rows in a set (0.02 sec)

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	2000-11-17	9586774748	Science
2	Deep Singh	M	98	1996-08-22	8988886577	Commerce
3	Ankit Sharma	M	76	2000-02-02	NULL	Science
4	Radhika Gupta	F	78	1996-12-03	9818675444	Humanities
6	Diksha Sharma	F	80	1999-12-17	9897666650	Humanities
7	Gurpreet Kaur	F	65	2000-01-04	7560875609	Science
8	Akshay Dureja	M	90	1997-05-05	9560567890	Commerce
10	Prateek Mittal	M	75	2000-12-25	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 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	1998-04-21	9845639990	Vocational
9	Shreya Anand	F	70	1999-10-08	NULL	Vocational

2 rows in a set (0.02 sec)

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 and VARCHAR. The LIKE operator searches the column to find if a 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	1996-08-22	8988886577	Commerce
6	Diksha Sharma	F	80	1999-12-17	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	Mobile_no	Stream
3	Ankit Sharma	M	76	NULL	Science
4	Radhika Gupta	F	78	9818675444	Humanities
6	Diksha Sharma	F	80	9897666650	Humanities
8	Akshay Dureja	M	90	9560567890	Commerce

4 rows in a set (0.02 sec)

The above command shall display the records for those students whose name ends with the 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	1996-08-22	8988886577	Commerce
5	Payal Goel	F	82	1998-04-21	9845639990	Vocational
7	Gurpreet Kaur	F	65	2000-01-04	7560875609	Science
8	Akshay Dureja	M	90	1997-05-05	9560567890	Commerce
9	Shreya Anand	F	70	1999-10-08	NULL	Vocational
10	Prateek Mittal	M	75	2000-12-25	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 the character 'e' anywhere in it.

```
mysql> SELECT * From Student WHERE Name LIKE "_e%";
```

Resultant table: Student

Rollno	Name	DOB
2	Deep Singh	1996-08-22

1 row in a set (0.02 sec)

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

NOT LIKE

The NOT LIKE operator works opposite to LIKE operator. It returns the rows that do not match the specified pattern.

For example,

```
mysql> SELECT Rollno, Name, Marks, DOB FROM Student WHERE Name
       NOT LIKE "%r_ _";
```

Resultant table: Student

Rollno	Name	Gender	Marks	DOB
1	Raj Kumar	M	93	2000-11-17
2	Deep Singh	M	98	1996-08-22
4	Radhika Gupta	F	78	1999-12-03
5	Payal Goel	F	82	1998-04-21
7	Gurpreet Kaur	F	65	2000-01-04
8	Akshay Dureja	M	90	1997-05-05
9	Shreya Anand	F	70	1999-10-08
10	Prateek Mittal	M	75	2000-12-25

8 rows in a set (0.02 sec)

This command shall display the Rollno, Name, Marks and DOB of all the students whose Name does not contain the letter 'r' from the third last position.

8.15.3 Comments in SQL

A comment is a text which is ignored by the SQL compiler and is not executed at all. It is given for documentation purpose only. A comment usually describes the purpose of the statement given within an application.

SQL Comments are used to understand the functionality of the program without looking into it. The comments give us an idea about what is written in the given SQL statement and how it works. The comments can make an application or code easier to read as well as to maintain. A comment can be placed between any keywords, parameters or punctuation marks in a statement. Comments can be either single-line comments or multiple-line comments.

SQL or MySQL supports three comment styles:

- **Comments beginning with -- (followed by a space):** The two dash lines indicate a single-line comment in SQL statements. These single-line comments are basically used to show the comments at the start and end of a program. A user can easily use this comment type to explain the flow of program. This text cannot extend to a new line and ends with a line break.
- **Comments beginning with #:** The comments begin with '#' symbol followed by the text to be displayed for the user's information. This text cannot extend to a new line and ends with a line break.
- **Comments beginning with /*:** Multi-line comments begin with a slash and an asterisk (/*) followed by the text of the comment. This text can span multiple lines. The comment ends with an asterisk and a slash (*/). The opening and terminating characters need not be separated from the text by a space or a line break.

For example,

```
mysql> SELECT Rollno, Name, Stream
    -> /* This statement shall display the records of all those students
       who are in Science stream and have secured marks more than 75. */
    -> FROM Student # student table in use
    -> WHERE Stream= 'Science' and Marks > 75;--condition for selection
```

8.16 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,

Table: Student

Student_name	Date_of_birth
Raj Kumar	2000-11-17
Deep Singh	1996-08-22
Ankit Sharma	2000-02-02
Radhika Gupta	1999-12-03
Payal Goel	1998-04-21
Diksha Sharma	1999-12-17
Gurpreet Kaur	2000-01-04
Akshay Dureja	1997-05-05
Shreya Anand	1999-10-08
Prateek Mittal	2000-12-25

```
mysql> SELECT Name AS "Student_name", DOB AS "Date_of_birth"
       FROM Student;
```

Alias name for fields, Name and DOB

10 rows in a set (0.02 sec)

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

For example,

```
mysql> SELECT 22/7 AS PI;
```

Output:

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

POINTS TO REMEMBER

- If the **alias_name** contains spaces, you must enclose it 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.

8.17 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,

```
mysql> 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	2000-11-17
2	Deep Singh	was born on	1996-08-22
3	Ankit Sharma	was born on	2000-02-02
4	Radhika Gupta	was born on	1999-12-03
5	Payal Goel	was born on	1998-04-21
6	Diksha Sharma	was born on	1999-12-17
7	Gurpreet Kaur	was born on	2000-01-04
8	Akshay Dureja	was born on	1997-05-05
9	Shreya Anand	was born on	1999-10-08
10	Prateek Mittal	was born on	2000-12-25

10 rows in a set (0.02 sec)

8.18 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;
```

Rollno	Name	Marks
8	Akshay Dureja	90
3	Ankit Sharma	76
2	Deep Singh	98
6	Diksha Sharma	80
7	Gurpreet Kaur	65
5	Payal Goel	82
10	Prateek Mittal	75
4	Radhika Gupta	78
1	Raj Kumar	93
9	Shreya Anand	70

- 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;
```

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

8.19 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 8.6: 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, salary, job and city) with the following structure:

Ecode	Name	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

- **MAX()**

  MAX() function is used to find the highest value among the given set of values of any column or expression based on the 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.

- **MIN()**

  MIN() function is used to find the lowest value among the given set of values of any column or expression based on the 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.

- **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;
```

Output:

```
+---------------+
| 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.

- **AVG()**

AVG() function is used to find the average value of any column or expression based on a column. Like sum(), 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 is not defined.

For example,

```
mysql> SELECT AVG(Salary) FROM EMPLOYEE;
```

Ecode	Name	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

Output:

```
+---------------+
| AVG(Salary) |
+---------------+
| 6166.66       |
+---------------+
```

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 6166.66.

- **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 the 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 along with NULL values, if any, in the table.

For example,

```
mysql> SELECT COUNT(*) FROM EMPLOYEE;
```

Output:

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

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

```
mysql> SELECT COUNT(DISTINCT City) FROM EMPLOYEE;
```

Output:

```
+-----------------------------+
| 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 table Employee given in the previous section with NULL values against the Salary field for some employees.

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 ignored.)

```
mysql> SELECT COUNT(Salary) FROM Employee;
```
Output: 3 (NULL values are 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.)

8.20 GROUP BY

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns. It groups the rows on the basis of the values present in one of the columns and then the aggregate functions are applied on any column of these groups to obtain the result of the query.

This clause can be explained with reference to the table student; the rows can be divided into four groups on the basis of the column Stream. One group of rows belongs to "Science" stream, another belongs to "Commerce" stream, the third group belongs to "Humanities" stream and the fourth belongs to "Vocational" stream. Thus, by using GROUP BY clause, the rows can be divided on the basis of the stream column.

Syntax for the GROUP BY clause is:

SELECT <column1, column2, ...column_n>, <aggregate_function (expression)>

FROM <tables>

WHERE <conditions>

GROUP BY <column1>, <column2>, ... <column_n>;

Here, *column_names* must include the columns on the basis of which grouping is to be done.

aggregate_function can be a function such as sum(), count(), max(), min(), avg(), etc.

For example, to display the name, stream, marks and count the total number of students who have secured more than 90 marks according to their stream.

```
mysql> SELECT Name, Stream, count(*) AS "Number of students"
       FROM Student
       WHERE Marks>90
       GROUP BY Stream;
```

Resultant table: Student

Name	Stream	Number of students	Marks
Raj Kumar	Science	1	93
Deep Singh	Commerce	2	98

2 rows in a set (0.02 sec)

8.21 HAVING CLAUSE

The HAVING clause is used in combination with the GROUP BY clause. It can be used in a SELECT statement to filter the records by specifying a condition which a GROUP BY returns.

The purpose of using HAVING clause with GROUP BY is to allow aggregate functions to be used along with the specified condition. This is because the aggregate functions are not allowed to be used with WHERE clause as it is evaluated on a single row whereas the aggregate functions are evaluated on a group of rows.

Thus, if any aggregate function is to be used after the FROM clause in a SELECT command, then instead of using the WHERE clause, HAVING clause should be used.

The Syntax for HAVING clause is:

SELECT <column1>, <column2>, ...<column_n>, <aggregate_function (expression)>

FROM <tables>

WHERE <condition/predicates>

GROUP BY [<column1, column2, ... column_n]

HAVING [<condition1 ... condition_n>];

For example,

```
mysql> SELECT Stream, SUM(Marks) AS "Total Marks"
       FROM Student
       GROUP BY Stream
       HAVING MAX(Marks) <85;
```

Stream	Total Marks
Science	151
Commerce	0
Humanities	158
Vocational	152

In the given output, the sum(Marks) for Commerce Stream is 0 since none of the values for field Marks in the Commerce Stream is less than 85.

CTM: SELECT statement can contain only those attributes which are already present in the GROUP BY clause.

8.22 AGGREGATE FUNCTIONS AND CONDITIONS ON GROUPS (HAVING CLAUSE)

You may use any condition on group, if required. HAVING <condition> clause is used to apply a condition on a group.

```
mysql> SELECT Job, SUM(Pay) FROM EMP GROUP BY Job HAVING SUM(Pay)>=8000;
mysql> SELECT Job, SUM(Pay) FROM EMP GROUP BY Job HAVING AVG(Pay)>=7000;
mysql> SELECT Job, SUM(Pay) FROM EMP GROUP BY Job HAVING COUNT(*)>=5;
mysql> SELECT Job, MIN(Pay), MAX(Pay), AVG(Pay) FROM EMP
       GROUP BY Job HAVING SUM(Pay)>=8000;
mysql> SELECT Job, SUM(Pay) FROM EMP WHERE City='Dehradun'
       GROUP BY Job HAVING COUNT(*)>=5;
```

WHERE clause works in respect to the whole table but **HAVING** clause works on Group only. If WHERE and HAVING both are used, then WHERE will be executed first. Where is used to put a condition on individual row of a table whereas HAVING is used to put a condition on an individual group formed by GROUP BY clause in a SELECT statement.

Aggregate Functions and Group (GROUP BY Clause): Other Combinations

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	Bengaluru
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.

An Aggregate function may be applied on a column with DISTINCT or * (ALL) symbol. If nothing is given, ALL scope is assumed.

- **Using sum (<Column>)**

  This function returns the sum of values in the given column or expression.

```
mysql> SELECT SUM(Salary) FROM Employee;
mysql> SELECT SUM(DISTINCT Salary) FROM Employee;
mysql> SELECT SUM(Salary) FROM Employee WHERE City='Kanpur';
mysql> SELECT SUM(Salary) FROM Employee GROUP BY City HAVING
       City='Kanpur';
mysql> SELECT Job, SUM(Salary) FROM Employee GROUP BY Job;
```

- **Using min (<Column>)**

  This function returns the Minimum value in the given column.

```
mysql> SELECT MIN(Salary) FROM Employee;
mysql> SELECT MIN(Salary) FROM Employee GROUP BY City HAVING City='Kanpur';
mysql> SELECT Job, MIN(Salary) FROM Employee GROUP BY Job;
```

- **Using max (<Column>)**

  This function returns the Maximum value in the given column.

```
mysql> SELECT MAX(Salary) FROM Employee;
mysql> SELECT MAX(Salary) FROM Employee WHERE City='Kanpur';
mysql> SELECT MAX(Salary) FROM Employee GROUP BY City HAVING City='Kanpur';
```

- **Using avg (<Column>)**

  This function returns the Average value in the given column.

```
mysql> SELECT AVG(Salary) FROM Employee;
mysql> SELECT AVG(Salary) FROM Employee GROUP BY City; HAVING
       City='Kanpur';
```

➢ **Using count (<*|Column>)**

This function returns the number of rows in the given column.

```
mysql> SELECT COUNT (*) FROM Employee;
mysql> SELECT COUNT(Salary) FROM Employee GROUP BY City;
mysql> SELECT COUNT(*), SUM(Salary) FROM Employee GROUP BY Job HAVING
       Job= "Manager";
```

8.23 SQL JOINS

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them. While querying for a join, more than one table is considered in FROM clause. The process/function of combining data from multiple tables is called a JOIN.

SQL can extract data from two or even more than two related tables by performing either a physical or virtual join on the tables using WHERE clause.

This is unlike cartesian product which make all possible combinations of tuples. While using the JOIN clause of SQL, we specify conditions on the related attributes of two tables within the FROM clause. Usually, such an attribute is the primary key in one table and foreign key in another table.

The types of SQL joins are as follows:

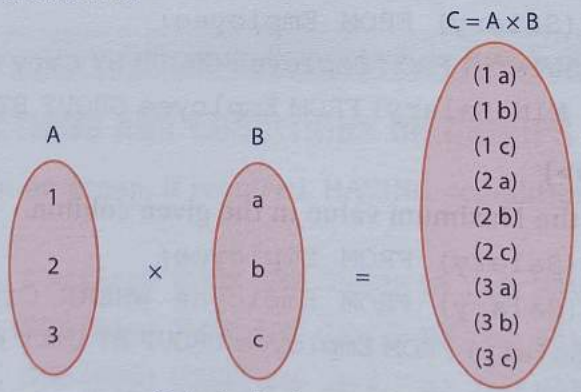
1. Cartesian Product (Cross Product)
2. Equi Join
3. Natural Join

Cartesian Product (Cross Product)

The Cartesian product is also termed as cross product or cross-join. The Cartesian product is a binary operation and is denoted by (×). The degree of the new relation formed is the sum of the degrees of two relations on which Cartesian product is performed. The number of tuples in the new relation is equal to the product of the number of tuples of the two tables on which Cartesian product is performed.

For example,

If A={1,2,3} and B={a,b,c}, find A × B.



Cartesian Product for A × B

Columns after Cartesian Product = 1+1 = 2

Rows after Cartesian Product = 3×3 = 9

Table: student

Rollno	Name
1	Rohan
2	Jaya
3	Teena

Table: games

gameno	gname
10	Football
11	Lawn tennis

Cartesian product for student × games:

```
mysql> SELECT Name, gname FROM Student, games;
```

student X games

Name	gname
Rohan	Football
Jaya	Football
Teena	Football
Rohan	Lawn Tennis
Jaya	Lawn Tennis
Teena	Lawn Tennis

Therefore, a Cartesian product is formed when no join conditions exist or are invalid. When we perform Cartesian product between two tables, all the rows in the first table are joined to all the rows in the second table. Using Cartesian product operation results in a large number of rows as the output, so it is seldom used.

Equi Join

An Equi join is a simple SQL join condition that uses the equal to sign (=) as a comparison operator for defining a relationship between two tables on the basis of a common field, *i.e.*, primary key and foreign key.

Syntax for Equi Join:

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

FROM <table1>, <table2>

WHERE <table1. Primary key column> = <table2.foreign key column>;

For example,

Table: Student

Rollno	Name
1	Rohan
2	Jaya
3	Teena
4	Diksha

Table: Fees

Rollno	Fee
4	4500
2	5500
3	5000

```
mysql> SELECT A.Rollno, A.Name, B.Fee FROM
       Student A, Fees B
       WHERE A.Rollno = B.Rollno;
```

Resultant Table

Rollno	Name	Fee
4	Diksha	4500
2	Jaya	5500
3	Teena	5000

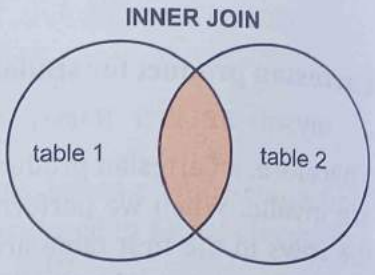
In the given SELECT statement, A and B are the alias names (alternate names) for the tables student and fees respectively. So, we can always use alternate name in place of primary name for the two tables to be joined.

Equi joins are further classified as:

1. Inner Join
2. Outer Join

1. **Inner Join/Intersection:** The Inner Join is a classification of equi join where either of the equivalent queries gives the intersection of two tables, *i.e.*, it returns the rows which are common in both the tables.

```
mysql> SELECT A.Rollno, A.Name, B.Fee FROM Student A, fees B
WHERE A.Rollno = B.Rollno ORDER BY A.Rollno;
```



Thus, the output for the given command will be:

Resultant Table

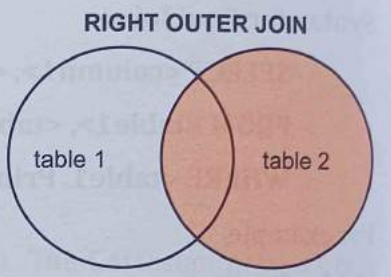
Rollno	Name	Fee
2	Jaya	5500
3	Teena	5000
4	Diksha	4500

3 rows in a set (0.00 sec)

2. **Outer Join:** Outer join is categorized as Left and Right Outer Join.

Right Outer Join: The Right Join keyword returns all rows from the right table (table 2), with the matching rows in the left table (table 1). The result is NULL in the left side when there is no match.

```
mysql> SELECT A.Rollno, A.Name, B.Fee
       FROM Student A RIGHT JOIN Fees B ON
       A.Rollno = B.Rollno ORDER BY B.Fee desc;
```



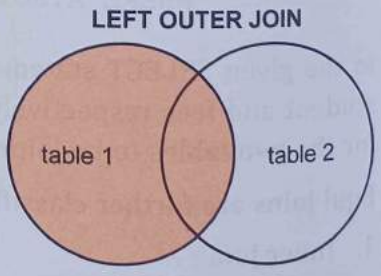
Resultant Table

Rollno	Name	Fee
2	Jaya	5500
3	Teena	5000
4	Diksha	4500

3 rows in a set (0.00 sec)

Left Outer Join: The Left Join keyword returns all rows from the left table (table 1), with the matching rows in the right table (table 2). The result is NULL in the right side when there is no match.

```
mysql> SELECT A.Rollno, Name, Fee
       FROM Student A LEFT JOIN Fees B
       ON A.Rollno = B.Rollno
```



Resultant Table

Rollno	Name	Fee
1	Rohan	NULL
2	Jaya	5500
3	Teena	5000
4	Diksha	4500

4 rows in a set (0.01 sec)

Let us take an example of two tables given below to explain the concept of MySQL Joins:

Table: Vehicle

CODE	VTYPE	PERKM
101	VOLVO BUS	160
102	AC DELUXE BUS	150
103	ORDINARY BUS	90
105	SUV	40
104	CAR	20

Note:

- PERKM is Freight Charges per kilometre
- VTYPE is Vehicle Type

Table: Travel

NO	NAME	TDATE	KM	CODE	NOP
101	Janish Kin	2015–11–13	200	101	32
103	Vedika Sahai	2016–04–21	100	103	45
105	Tarun Ram	2016–03–23	350	102	42
102	John Fen	2016–02–13	90	102	40
107	Ahmed Khan	2015–01–10	75	104	2
104	Raveena	2016–05–28	80	105	4
106	Kripal Anya	2016–02–06	200	101	25

Note:

- NO is Traveller Number
- KM is Kilometres travelled
- NOP is number of travellers in vehicle
- TDATE is Travel Date

(a) To display NO, NAME, TDATE from the table TRAVEL in descending order of NO.

Ans.
```
SELECT NO, NAME, TDATE FROM Travel ORDER BY NO DESC;
```

(b) To display the NAME of all the travellers from the table TRAVEL who are travelling by vehicle with code 101 or 102.

Ans.
```
SELECT NAME FROM Travel
WHERE CODE='101' OR
CODE='102';
```

OR

```
SELECT NAME FROM Travel
WHERE CODE IN('101','102');
```

(c) To display the NO and NAME of those travellers from the table TRAVEL who travelled between '2015-04-01' and '2015-12-31'.

Ans.
```
SELECT NO, NAME FROM Travel
WHERE TDATE >= '2015/04/01' and TDATE <= '2015/12/31';
```

OR

```
SELECT NO, NAME FROM Travel
WHERE TDATE BETWEEN '2015/04/01' and '2015/12/31';
```

(d) To display the CODE, NAME, VTYPE from both the tables with distance travelled (KM) is less than 90 km.

Ans.
```
SELECT A.CODE, NAME,
VTYPE FROM TRAVEL A,
VEHICLE B
WHERE A.CODE=B.CODE AND KM<90;
```

JOIN Operation using Table alias names

(e) To display the NAME and amount to be paid for vehicle code as 105. Amount to be paid is calculated as the product of KM and PERKM.

Ans.
```
SELECT NAME,
KM*PERKM FROM TRAVEL A,
VEHICLE B
WHERE A.CODE=B.CODE AND A.CODE='105';
```

Natural Join

The JOIN in which only one of the identical columns exists is called Natural Join. It is similar to Equi Join except that duplicate columns are eliminated in Natural Join that would otherwise appear in Equi Join.

In Natural Join, we can also specify the names of columns to fetch in place of (*), which is responsible for the appearance of common column twice in the output. In other words, the SQL Natural Join is a type of Equi Join and is structured in such a way that columns with the same name of associated tables will appear only once. The following two conditions are must for Natural Join:

- The associated tables have one or more pairs of identically named columns.
- The columns must be of same data type.

In NATURAL JOIN, the join condition is not required; it automatically joins based on the common column value.

Natural Join is an alternative method to Equi Join.

Syntax is:

```
SELECT * FROM <Table1> NATURAL JOIN <Table2>;
```

For example,

```
mysql>SELECT * FROM Student NATURAL JOIN Fees;
```

Resultant Table

Rollno	Name	Fee
4	Diksha	4500
2	Jaya	5500
3	Teena	5000

3 rows in a set (0.00 sec)

MEMORY BYTES

- 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 Query Language (DQL).
- DDL is used to create and delete tables, databases, views, etc.
- DML is used to modify and update the database.
- DESCRIBE or DESC is used to see 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.
- 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 the range of values that the column values must fall within to make the condition true.
- The IN operator selects values that match any value in the given list of values.
- % and _ are two wild card characters. The percent (%) 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 INTO statement is used for inserting new rows or data into an existing table.
- The ORDER BY keyword in MySQL is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
- HAVING clause is used in combination with GROUP BY clause.
- GROUP BY clause is used whenever aggregate functions by group are required.
- The HAVING clause is used to place conditions on groups created by GROUP BY clause because here the WHERE clause is not usable.
- An aggregate function is a function where the values of multiple rows are grouped together as input based on certain criteria to form a single value of more significant meaning.
- The COUNT() function returns the number of rows that matches a specified criteria.
- The AVG() function returns the average value of a numeric column.
- The SUM() function returns the total sum of a numeric column.

OBJECTIVE TYPE QUESTIONS

1. Fill in the blanks.
   (a) ..... is a freely-available open-source RDBMS that implements SQL.
   (b) MySQL provides a dummy table named ..... .

(c) The ..... keyword eliminates duplicate records from the results of a SELECT statement.

(d) Patterns in MySQL are described using two special wild card characters such as ..... and ..... .

(e) The keyword ..... is used to select rows that do not match the specified pattern of characters.

(f) The default order of ORDER BY clause is ..... .

(g) The ..... function is used to count the number of records in a column.

(h) The rows of the table (relation) are referred to as ..... .

(i) The non-key attribute which helps to make relationship between two tables is known as ..... .

(j) To specify filtering condition for groups, the ..... clause is used in MySQL.

2. State whether the following statements are True or False.

(a) Duplication of data is known as Data Redundancy.

(b) An Attribute is a set of values of a dissimilar type of data.

(c) MySQL supports different platforms like UNIX and Windows.

(d) UPDATE TABLE command is used to create table in a database.

(e) Null (unavailable and unknown) values are entered by the following command:
INSERT INTO TABLE_NAME VALUES ("NULL");

(f) ALTER TABLE command is used to modify the structure of the table.

(g) Each SQL table must have at least one column and one row.

(h) Foreign key column derives its value from the primary key of the parent table.

(i) DISTINCT clause is used to remove redundant rows from the result of the SELECT statement.

(j) SELECT MIN (salary) FROM Employee will return the highest salary from the table.

(k) Group functions can be applied to any numeric values, some text types and DATE values.

3. Multiple Choice Questions (MCQs)

(a) The ..... allows us to perform tasks related to data definition.
(i) DDL (ii) DML (iii) TCL (iv) None of these

(b) The ..... allows us to perform tasks related to data manipulation.
(i) DDL (ii) DML (iii) TCL (iv) None of these

(c) A ..... is a text that is not executed.
(i) Statement (ii) Query (iii) Comment (iv) Clause

(d) ..... are words that have a special meaning in SQL.
(i) Keyword (ii) Literal (iii) Variable (iv) Table

(e) Which command helps to open the database for use?
(i) Use (ii) Open (iii) Distinct (iv) Select

(f) Which of these commands helps to fetch data from relation?
(i) Use (ii) Show (iii) Fetch (iv) Select

(g) The ..... keyword eliminates duplicate rows from the results of a SELECT statement.
(i) OR (ii) DISTINCT (iii) ANY (iv) ALL

(h) ..... command helps to see the structure of a table/relation.
(i) Show (ii) Select (iii) Describe (iv) Order by

(i) ..... is known as range operator in MySQL.
(i) IN (ii) DISTINCT (iii) IS (iv) BETWEEN

(j) The ..... clause allows sorting of query results by one or more columns.
(i) ALL (ii) DISTINCT (iii) GROUP BY (iv) ORDER BY

(k) Which clause is used in query to place the condition on groups in MySQL?
(i) WHERE (ii) HAVING (iii) GROUP BY (iv) Both (i) & (ii)

(l) Which of the following is a DDL command?
(i) SELECT (ii) ALTER (iii) INSERT (iv) UPDATE

(m) Which of the following types of table constraints will prevent the entry of duplicate rows?
(i) Unique (ii) Distinct (iii) Primary Key (iv) NULL

SOLVED QUESTIONS

1. What is SQL?

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

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

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

Record: A record is a 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 records of salary respectively.

3. 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.

4. In SQL, name the clause that is used to display the tuples in ascending order of an attribute.

Ans. ORDER BY

5. What is NULL value?

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

6. Differentiate between WHERE and HAVING clause.

Ans. WHERE clause is used to select particular rows that satisfy a condition whereas HAVING clause is used to place condition on an individual group formed with GROUP BY clause.

For example, `SELECT * from Student WHERE Marks>75;`

This statement shall display the records for all the students who have scored more than 75 marks.

On the contrary, the statement—

`SELECT * FROM Student GROUP BY Stream HAVING Marks>75;`

shall display the records of all the students grouped together on the basis of stream but only for those students who have scored marks more than 75.

7. What is a NOT NULL constraint?

Ans. By default, we can add NULL values to the table columns until and unless NOT NULL constraint has been explicitly defined. If you do not want a column to have a NULL value, then you need to define that column with NOT NULL constraint.

8. What is the purpose of GROUP BY clause?

Ans. GROUP BY clause is used in a SELECT statement in combination with aggregate functions to group the result based on distinct values in a column.

9. What is HAVING clause?

Ans. HAVING clause is used in combination with GROUP BY clause in a SELECT statement to put condition on groups.

10. (a) What do you mean by Primary Key? Give a suitable example of a 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.

(b) Write a query that displays city, salesman name, code and commission from salesman table.

Ans.
```
SELECT City, Salesman_name, Code, Commission FROM Salesman;
```

(c) 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;
```

(d) Write a command that deletes all orders for the customer name SOHAN from table customer.

Ans.
```
DELETE FROM Customer WHERE Customer_name like 'SOHAN';
```
OR
```
DELETE FROM Customer WHERE customer_name = 'SOHAN';
```

(e) Differentiate between DROP and DELETE command.

Ans.

S.No.	DROP command	DELETE command
1	Drop command is used to delete a table structure along with all the records stored in it.	Delete command is used to delete all the records or some records from a table without deleting the table.
2	It is a DDL (Data Definition Language) command.	It is a DML (Data Manipulation Language) command.
3	It frees the memory occupied by the table.	The memory occupied by the table is not free even if we delete all the rows of the table.

(f) 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;
```

(g) 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 along with data types are given, which are separated by commas, while giving with ADD clause. 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);
```

(h) What are JOINS?

Ans. A JOIN is a query that combines tuples from more than one table. In a join query, the table names are given with FROM clause, separated by a comma. For example,

```
SELECT Name, Salary FROM Emp1, Emp2;
```

In this statement, the two tables are Emp1 and Emp2 from which the column name and salary are extracted.

(i) Differentiate between Alter and Update command.

Ans. Alter Command:

- It is used to change the columns of the existing table such as:
  - Adding a new column
  - Deleting a column

- Changing the data type of the column
- Renaming a column

■ It is a DDL command.

■ It is also used to add or delete the constraints on an existing table.

Update Command:

- It is used to change records of the table specified by a condition.
- It is a DML command.

11. 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;
```

12. Consider a database LOANS with the following table:

Table: LOANS

AccNo	Cust_Name	Loan_Amount	Instalments	Int_Rate	Start_Date	Interest
1	R.K. Gupta	300000	36	12.00	2009-07-19	1200
2	S.P. Sharma	500000	48	10.00	2008-03-22	1800
3	K.P. Jain	300000	36	NULL	2007-03-08	1600
4	M.P. Yadav	800000	60	10.00	2008-12-06	2250
5	S.P. Sinha	200000	36	12.50	2010-01-03	4500
6	P. Sharma	700000	60	12.50	2008-06-05	3500
7	K.S. Dhall	500000	48	NULL	2008-03-05	3800

Answer the following questions.

(i) Display the sum of all Loan Amounts whose Interest rate is greater than 10.

(ii) Display the Maximum Interest from Loans table.

(iii) Display the count of all loan holders whose name ends with 'Sharma'.

(iv) Display the count of all loan holders whose Interest rate is Null.

(v) Display the Interest-wise details of Loan Account Holders.

(vi) Display the Interest-wise details of Loan Account Holders with at least 10 instalments remaining.

Ans. (i)
```
Mysql> SELECT SUM(Loan_Amount) FROM LOANS WHERE Int_Rate >10;
```
(ii)
```
Mysql> SELECT MAX(Interest) FROM LOANS;
```
(iii)
```
Mysql> SELECT COUNT(*) FROM LOANS WHERE Cust_Name LIKE '%Sharma';
```
(iv)
```
Mysql> SELECT COUNT(*) FROM LOANS WHERE Int_Rate IS NULL;
```
(v)
```
Mysql> SELECT * FROM LOANS GROUP BY Interest;
```
(vi)
```
Mysql> SELECT * FROM LOANS GROUP BY Interest HAVING Instalments>=10;
```

[CBSE D 2016]

13. Consider the table 'HOTEL' given below:

EMPID	CATEGORY	SALARY
E101	MANAGER	60000
E102	EXECUTIVE	65000
E103	CLERK	40000
E104	MANAGER	62000
E105	EXECUTIVE	50000
E106	CLERK	35000

Mr. Vinay wanted to display average salary of each category. He entered the following SQL statement. Identify error(s) and rewrite the correct SQL statement.

```
SELECT CATEGORY, AVERAGE(SALARY) FROM HOTEL GROUP BY CATEGORY;
```

Ans.
```
SELECT CATEGORY, AVG(SALARY) FROM HOTEL GROUP BY CATEGORY;
```

14. What is an ORDER BY clause and GROUP BY clause?

Ans. ORDER BY clause is used to display the result of a query in a specific order (sorted order). The sorting can be done in ascending or in descending order. However, the actual data in the database is not sorted but only the results of the query are displayed in sorted order. If order is not specified then, by default, the sorting will be performed in ascending order.

For example,

```
SELECT Name, City FROM Student ORDER BY Name;
```

The above query returns name and city columns of table student sorted by name in ascending order.

For example,

```
SELECT * FROM Student ORDER BY City DESC;
```

It displays all the records of table student ordered by city in descending order.

GROUP BY clause

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

For example,

```
SELECT Name, COUNT(*) AS "Number of employees"
FROM Employee WHERE Salary>35000.
GROUP BY City;
```

It displays name and the total number of employees of each city who are getting salary greater than 35000.

15. Consider the following tables Product and Client. Write SQL commands for the statements (i) to (iii) and give outputs for SQL queries (iv) to (vi).

Table: PRODUCT

P_ID	Product Name	Manufacturer	Price
TP01	Talcum Powder	LAK	40
FW05	Face Wash	ABC	45
BS01	Bath Soap	ABC	55
SH06	Shampoo	XYZ	120
FW12	Face Wash	XYZ	95

Table: CLIENT

C_ID	Client Name	City	P_ID
01	Cosmetic Shop	Delhi	FW05
06	Total Health	Mumbai	BS01
12	Live Life	Delhi	SH06
15	Pretty Woman	Delhi	FW12
16	Dreams	Bengaluru	TP01

(i) To display the details of those Clients whose city is Delhi.

(ii) To display the details of Products whose Price is in the range of 50 to 100 (both values included).

(iii) To display the details of those products whose name ends with 'Wash'.

(iv) `SELECT DISTINCT City FROM CLIENT;`

(v) `SELECT Manufacturer, MAX(Price), MIN(Price), COUNT(*) FROM PRODUCT GROUP BY Manufacturer;`

(vi) `SELECT Product Name, Price * 4 FROM PRODUCT;`

Ans. (i) `SELECT * FROM CLIENT WHERE City="Delhi";`

(ii) `SELECT * FROM PRODUCT WHERE Price BETWEEN 50 and 100;`

(iii) `SELECT * FROM PRODUCT WHERE ProductName LIKE '%Wash';`

(iv)

City
Delhi
Mumbai
Bengaluru

(v)

Manufacturer	Max(Price)	Min(Price)	Count(*)
LAK	40	40	1
ABC	55	45	2
XYZ	120	95	2

(vi)

Product Name	Price*4
Talcum Powder	160
Face Wash	180
Bath Soap	220
Shampoo	480
Face Wash	380

16. 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).

17. 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 data accuracy and reliability in the database.
Following are the most commonly used constraints available in SQL:

(a) NOT NULL Constraint: Ensures that a column cannot have NULL value.
(b) DEFAULT Constraint: Provides a default value for a column when no value is specified.
(c) 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.
(d) PRIMARY Key: Uniquely identifies each row/record in a database table.
(e) FOREIGN Key: Uniquely identifies a row/record in any other database table.
(f) 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.

18. Write the outputs of the SQL queries (i) to (iii) & queries for (iv) to (viii) based on the relations Teacher and Posting given below: [CBSE Sample Paper 2020-21]

Table: Teacher

T_ID	Name	Age	Department	Date_of_join	Salary	Gender
1	Jugal	34	Computer Sc	2017-01-10	12000	M
2	Sharmilla	31	History	2008-03-24	20000	F
3	Sandeep	32	Mathematics	2016-12-12	30000	M
4	Sangeeta	35	History	2015-07-01	40000	F
5	Rakesh	42	Mathematics	2007-09-05	25000	M
6	Shyam	50	History	2008-06-27	30000	M
7	Shiv Om	44	Computer Sc	2017-02-25	21000	M
8	Shalakha	33	Mathematics	2018-07-31	20000	F

Table: Posting

P_ID	Department	Place
1	History	Agra
2	Mathematics	Raipur
3	Computer Science	Delhi

(i) `SELECT Department, COUNT(*) FROM Teacher GROUP BY Department;`

Ans.

Department	Count(*)
History	3
Computer Sc	2
Mathematics	3

(ii) `SELECT MAX(Date_of_Join),MIN(Date_of_Join) FROM Teacher;`

Ans.

MAX(Date_of_Join)	MIN(Date_of_Join)
2018-07-31	2007-09-05

(iii)
```
SELECT Teacher.Name,Teacher.Department, Posting.Place FROM Teacher, Posting
WHERE Teacher.Department = Posting.Department AND Posting.Place="Delhi";
```

Ans.

Name	Department	Place
Jugal	Computer Sc	Delhi
Shiv Om	Computer Sc	Delhi

(iv) To show all information about the teachers of History department.

Ans. `SELECT * FROM Teacher WHERE Department= "History";`

(v) To list the names of female teachers who are in Mathematics department.

Ans. `SELECT Name FROM Teacher WHERE Department= "Mathematics" AND Gender= "F";`

(vi) To list the names of all teachers with their date of joining in ascending order.

Ans. `SELECT Name FROM Teacher ORDER BY Date_of_join;`

(vii) To display teacher's name, salary, age for male teachers only.

Ans. `SELECT Name, Salary, Age FROM Teacher WHERE Gender='M';`

(viii) To display name, bonus for each teacher where bonus is 10% of salary.

Ans. `SELECT Name, Salary*0.1 AS Bonus FROM Teacher;`

19. Consider the following tables: COMPANY and MODEL.

Table: Company

Comp_ID	CompName	CompHO	ContactPerson
1	Titan	Okhla	C.B. Ajit
2	Ajanta	Najafgarh	R. Mehta
3	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 (iv) and output for (v) and (vi).

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

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

(iii) To display the Model_ID, Comp_ID, Cost from the table Model, CompName and ContactPerson from Company table, with their corresponding Comp_ID.

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

(v) `SELECT COUNT(DISTINCT CompHO) FROM Company;`

(vi) `SELECT CompName, 'Mr.', ContactPerson FROM Company WHERE CompName LIKE '%a';`

Ans. (i)
```
SELECT * FROM Model
ORDER BY DateOfManufacture;
```

(ii)
```
SELECT * FROM Model
WHERE year(DateOfManufacture) = 2011 AND Cost < 2000;
```

(iii)
```
SELECT Model_ID, Comp_ID, Cost, CompName, ContactPerson FROM Model, Company
WHERE Model.Comp_ID = Company.Comp_ID;
```

(iv)
```
UPDATE Model
SET Cost = Cost - 0.15*Cost;
```

(v)

count(distinct CompHO)
3

(vi)

CompName	Mr.	ContactPerson
Ajanta	Mr.	R. Mehta
Maxima	Mr.	B. Kohli

20. 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 (iv) and output for (v) and (vi):

(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) To display C_ID, ClientName, City of all the clients (including the ones that have not purchased a product) and their corresponding ProductName sold.

(v) `SELECT COUNT(DISTINCT Manufacturer) FROM Product;`

(vi) `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)
```
SELECT C_ID, ClientName, City, ProductName FROM Client LEFT JOIN Product
ON Client. P_ID = Product.P_ID;
```

(v)

COUNT(DISTINCT Manufacturer)
3

(vi)

C_ID	Client_Name	City
6	Total Health	Mumbai

21. 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 (iv) and output for (v) and (vi):

(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 display ConsumerName, Address from the table Consumer and Company and Price from Stationery table, with their corresponding S_ID.

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

(v) `SELECT COUNT(DISTINCT Address) FROM Consumer;`

(vi)
```
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)
```
SELECT ConsumerName, Address, Company, Price FROM Consumer,
Stationery WHERE Consumer.P_ID= Stationery.S_ID;
```

(iv)
```
UPDATE Stationery
SET Price = Price + 2;
```

(v)

COUNT(DISTINCT Address)
3

(vi)

StationeryName	price * 3
Pencil	15

22. 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 (iv) and output for (v) and (vi):

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

(iv) To display Dcode, Dname from Dealer table and Item, UnitPrice from Stock table of all the Dealers (including the dealer details that have not sold any item)

(v) `SELECT COUNT(DISTINCT Dcode) FROM Stock;`

(vi) `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)
```
SELECT Dealer.Dcode, Dname, Item, UnitPrice FROM Dealer LEFT JOIN Stock ON
Dealer.Dcode = Stock.Dcode;
```

(v)

COUNT(DISTINCT Dcode)
3

(vi)

Qty * UnitPrice
4400

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

Note: Answer 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. It is obtained by adding the columns and multiplying the rows of two tables.

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 Maximum and Minimum 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 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(DISTINCT Area)
3

24. (a) Define a candidate key with example. Write SQL queries for (b) to (f) and write the output for the SQL queries mentioned in parts (g1) to (g3) on the basis of tables PRODUCTS and SUPPLIERS.

Table: PRODUCTS

PID	PName	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

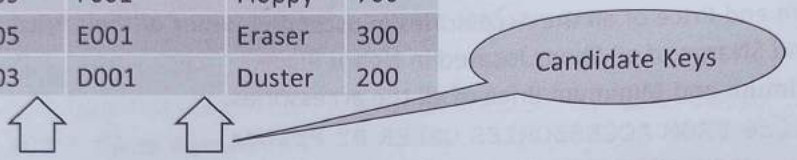
Table: SUPPLIERS

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; such attribute(s) is/are eligible for becoming a Primary key and is/are known as Candidate Keys. For example, in the table Item, columns Ino and ICode are eligible for becoming a Primary key and, thus, can be classified as candidate keys.

Table: Item

Ino	ICode	Item	QTY
101	P001	Pen	560
102	PE01	Pencil	780
104	C001	CD	450
109	F001	Floppy	700
105	E001	Eraser	300
103	D001	Duster	200



(b) To arrange and display all the records of table Products on the basis of product name in the ascending order.

Ans.
```
SELECT * FROM PRODUCTS ORDER BY PNAME;
```

(c) 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 AND PRICE<=15000;
```
OR
```
SELECT PNAME, PRICE FROM Products WHERE Price BETWEEN 10000 AND 15000;
```

(d) To display the price, product name and quantity (*i.e.*, qty) of those products which have quantity more than 100.

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

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

Ans.
```
SELECT SNAME FROM SUPPLIERS WHERE CITY="DELHI" OR CITY="CHENNAI";
```

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

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

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

(g1)
```
SELECT DISTINCT SUPCODE FROM PRODUCTS;
```
(g2)
```
SELECT MAX(PRICE), MIN(PRICE) FROM PRODUCTS;
```
(g3)
```
SELECT PRICE*QTY AS "AMOUNT" FROM PRODUCTS WHERE PID=104;
```

Ans. (g1)

DISTINCT SUPCODE
S01
S02
S03

(g2)

MAX (PRICE)	MIN (PRICE)
28000	1100

(g3)

AMOUNT
550000

25. (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 a database table. It cannot accept null, duplicate values.

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

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

Table: Student Registration

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 table CARDEN given below:

Table: CARDEN

Ccode	CarName	Make	Color	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

Write SQL commands for the following statements:

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

(ii) To display 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;
```

(c) 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

26. Consider the tables EMPLOYEE and SALGRADE given below and answer (a) and (b) parts of this question.

Table: EMPLOYEE

ECODE	NAME	DESIG	SGRADE	DOJ	DOB
101	Abdul Ahmad	EXECUTIVE	S03	2003-03-23	1980-01-13
102	Ravi Chander	HEAD-IT	S02	2010-02-12	1987-06-22
103	John Ken	RECEPTIONIST	S03	2009-06-24	1983-02-24
105	Nazar Ameen	GM	S02	2006-08-11	1984-03-03
108	Priyam Sen	CEO	S01	2004-12-29	1982-01-19

Table: SALGRADE

SGRADE	SALARY	HRA
S01	56000	18000
S02	32000	12000
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 in between '09-Feb-2006' and '08-Aug-2009'.

(iv) To add a new row with the following content:
109,'Harish Roy','HEAD-IT','S02','9-Sep-2007','21-Apr-1983'

Ans. (i)
```
SELECT * FROM EMPLOYEE ORDER BY DOJ DESC;
```
(ii)
```
SELECT NAME, DESIG FROM EMPLOYEE WHERE SGRADE='S02' OR SGRADE='S03';
```
(iii)
```
SELECT * FROM EMPLOYEE WHERE DOJ BETWEEN '2006-02-09' AND '2009-08-02';
```
(iv)
```
INSERT INTO EMPLOYEE VALUES(109, 'Harish Roy', 'HEAD-IT', 'S02',
'2007-09-09', '1983-04-21');
```

(b) 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)

MIN (DOB)	MAX (DOJ)
1980-01-13	2010-02-12

(iii)

SGRADE	SALARY+HRA
S02	44000

27. Consider the following tables STOCK and DEALERS and answer (a) and (b) parts of this question:

Table: STOCK

ItemNo	Item	Dcode	Qty	UnitPrice	StockDate
5005	Ball Pen 0.5	102	100	16	2010-03-31
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	2009-01-01
5001	Eraser Small	102	210	5	2009-03-19
5004	Eraser Big	102	60	10	2009-12-12
5009	Sharpener Classic	103	160	8	2009-01-23

Table: DEALERS

Dcode	Dname
101	Reliable Stationers
103	Classic Plastics
102	Clear Deals

(a) Write SQL commands for the following statements:

(i) 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;
```

(ii) 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;
```

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

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

Ans.

Qty*UnitPrice
4400

(ii)
```
SELECT MIN(StockDate) FROM Stock;
```

Ans.

MIN (StockDate)
2009-01-01

28. Write SQL commands for (a) to (f) on the basis of table SPORTS:

Table: SPORTS

StudentNo	Class	Name	Game1	Grade1	Game2	Grade2
10	7	Sameer	Cricket	B	Swimming	A
11	8	Sujit	Tennis	A	Skating	C
12	7	Kamal	Swimming	B	Football	B
13	7	Venna	Tennis	C	Tennis	A
14	9	Archana	Basketball	A	Cricket	A
15	10	Arpit	Cricket	A	Athletics	C

(a) Display the names of the students who have grade 'C' in either Game1 or Game2 or both.
(b) Display the number of students getting grade 'A' in Cricket.
(c) Display the names of the students who have the same game for both Game1 and Game2.
(d) Display the game taken up by the students, whose name starts with 'A'.

(e) Add a new column named 'Marks'.

(f) Assign a value 200 for marks for all those who are getting grade 'B' or grade 'A' in both Game1 and Game2.

Ans. (a)
```
SELECT Name FROM SPORTS WHERE Grade1='C' OR Grade2='C';
```
(b)
```
SELECT COUNT(*) FROM SPORTS WHERE (Grade1='A' AND Grade2='A')
AND (Game1='Cricket' OR Game2='Cricket');
```
(c)
```
SELECT Name FROM SPORTS WHERE Game1 = Game2;
```
(d)
```
SELECT Game1, Game2 FROM SPORTS WHERE Name LIKE 'A%';
```
(e)
```
ALTER TABLE SPORTS ADD (Marks INT(4));
```
(f)
```
UPDATE SPORTS SET Marks=200 WHERE (Grade1='A' OR Grade2='A')
OR (Grade1='B' OR Grade2='B');
```

29. (a) Study the following table and write SQL queries for questions (i) to (iv) and output for (v) and (vi).

Table: Orders

Orderid	Pname	Quantity	Rate	Sale_date	Discount
1001	Pen	10	20	2019-10-05	NULL
1002	Pencil	20	10	2019-10-21	NULL
1003	Book	10	100	2019-11-02	50
1004	Eraser	100	5	2019-12-05	25
1005	Notebook	50	20	2019-12-10	NULL

(i) Write SQL query to display Pname, Quantity and Rate for all the orders that are either Pencil or Pen.

(ii) Write SQL query to display the orders which are not getting any Discount.

(iii) Write SQL query to display the Pname, Quantity and Sale_date for all the orders whose total cost (Quantity * Rate) is greater than 500.

(iv) Write SQL query to display the orders whose Rate is in the range 20 to 100.

(v)
```
SELECT Pname, Quantity FROM Orders WHERE Pname LIKE('_e%');
```
(vi)
```
SELECT Pname, Quantity, Rate FROM Orders ORDER BY Quantity desc;
```

Ans. (i)
```
SELECT Pname, Quantity, Rate FROM Orders WHERE Pname IN ('Pencil','Pen');
```
(ii)
```
SELECT * FROM Orders WHERE Discount IS NULL;
```
(iii)
```
SELECT Pname, Quantity, Sale_date FROM Orders WHERE Quantity * Rate > 500;
```
(iv)
```
SELECT * FROM Orders WHERE Rate BETWEEN 20 AND 100;
```
(v)

Pname	Quantity
Pen	10
Pencil	20

(vi)

Pname	Quantity	Rate
Eraser	100	5
Notebook	50	20
Pencil	20	20
Book	10	100
Pen	10	20

(b) Based on the Orders Table, write SQL query to display the minimum quantity, maximum quantity and total quantity of Orders.

Ans.
```
SELECT MIN(Quantity), MAX(Quantity), SUM(Quantity) FROM Orders;
```

(c) What is the difference between COUNT(*) and COUNT(Column Name)?

Ans. COUNT(*) counts total number of rows in a table including the rows that contain the NULL values. COUNT(Column name) ignores the cells (row values) in the column containing NULL values.

UNSOLVED QUESTIONS

1. What is an Alternate Key?
2. What are views? How are they useful?
3. Define the following terms:
   (a) Relation
   (b) Tuple
   (c) Attribute
   (d) Domain
4. What do you understand by the terms candidate key and alternate key in a relational database?
5. What is SQL? What are different categories of commands available in SQL?
6. What is a database system? What is its need?
7. Differentiate between DDL and DML commands.
8. Give the terms for each of the following:
   (a) Collection of logically related records.
   (b) DBMS creates a file that contains description about the data stored in the database.
   (c) Attribute that can uniquely identify the tuples in a relation.
   (d) Special value that is stored when actual data value is unknown for an attribute.
   (e) An attribute which can uniquely identify tuples of the table but is not defined as primary key of the table.
   (f) Software that is used to create, manipulate and maintain a relational database.
9. An organization wants to create two tables EMP & DEPENDENT to maintain the following details about its employees and their dependents.
   EMPLOYEE(AadharNumber, Name, Address, Department, EmployeeID)
   DEPENDENT(EmployeeID, DependentName, Relationship)
   (a) Name the attributes of EMPLOYEE which can be used as candidate keys.
   (b) The company wants to retrieve details of dependents of a particular employee. Name the tables and the key which are required to retrieve these details.
   (c) What is the degree of EMPLOYEE and DEPENDENT relation?
10. What is a data type? Name some data types available in MySQL.
11. Differentiate between char and varchar data types.
12. Which operator concatenates two strings in a query result?
13. How would you calculate 13*15 in SQL?
14. Which keywords eliminate redundant data from a query?
15. What is the significance of GROUP BY clause in an SQL query?
16. What is the difference between WHERE and HAVING clause in SQL select command?
17. 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.
18. Define the following terms:
   (i) Database
   (ii) Data Inconsistency
   (iii) Primary Key
   (iv) Candidate Key
   (v) Foreign Key

19. Consider the following EMP and DEPT tables:

Table: EMP

EmpNo	EmpName	City	Designation	DOJ	Sal	Comm	DeptID
8369	SMITH	Mumbai	CLERK	1990-12-18	800.00	NULL	20
8499	ANYA	Varanasi	SALESMAN	1991-02-20	1600.00	300.00	30
8521	SETH	Jaipur	SALESMAN	1991-02-22	1250.00	500.00	30
8566	MAHADEVAN	Delhi	MANAGER	1991-04-02	2985.00	NULL	20

Table: DEPT

DeptID	DeptName	MgrID	Location
10	SALES	8566	Mumbai
20	PERSONNEL	9698	Delhi
30	ACCOUNTS	4578	Delhi
40	RESEARCH	8839	Bengaluru

Write the SQL command to get the following:

(a) Show the minimum, maximum and average salary of managers.
(b) Count the number of clerks in the organization.
(c) Display the designation-wise list of employees with name, salary and date of joining.
(d) Count the number of employees who are not getting commission.
(e) Show the average salary for all departments with more than 5 working people.
(f) List the count of employees grouped by DeptID.
(g) Display the maximum salary of employees in each department.
(h) Display the name of employees along with their designation and department name.
(i) Count the number of employees working in ACCOUNTS department.

20. Consider the following structure of TEACHER and STUDENT table:

Table: TEACHER

TeacherID	TName	City	Subject	Qualification	Designation	Pay

Table: STUDENT

StdID	Name	FName	Stream	TeacherID

Write the SQL commands to get the following:

(a) Show the names of students enrolled in the Science Stream.
(b) Count the number of students in the Commerce Stream.
(c) Count the number of teachers in each designation.
(d) Display the maximum pay of the teacher who is teaching English.
(e) Write a query to find out the number of students in each Stream in STUDENT table.

21. Consider the following tables STORE and SUPPLIERS. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).

Table: STORE

ItemNo	Item	Scode	Qty	Rate	LastBuy
2005	Sharpener Classic	23	60	8	2019-06-20
2003	Ball Pen 0.25	22	50	25	2020-02-02
2002	Gel Pen Premium	21	150	12	2000-02-24
2006	Gel Pen Classic	21	250	20	2019-03-11
2001	Eraser Small	22	220	6	2019-01-19
2004	Eraser Big	22	110	8	2019-12-02
2009	Ball Pen 0.5	21	180	18	2019-11-03

Table: SUPPLIERS

Scode	Sname
21	Premium Stationery
23	Soft Plastics
22	Tetra Supply

(i) To display details of all the items in the Store table in ascending order of LastBuy.
(ii) To display Itemno and item name of those items from Store table whose rate is more than 15 rupees.
(iii) To display the details of those items whose supplier code is 22 or quantity in store is more than 110 from the table Store.
(iv) To display minimum rate of items for each Supplier individually as per Scode from the table Store.
(v) `SELECT COUNT(DISTINCT Scode) FROM STORE;`
(vi) `SELECT Rate*Qty FROM STORE WHERE Itemno=2004;`
(vii) `SELECT Item, Sname FROM STORE S, SUPPLIER P WHERE S.Scode=P.Scode AND ItemNo=2006;`
(viii) `SELECT MAX(LastBuy) FROM STORE;`

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

BOOKS

Book_ID	Book_name	author_name	Publishers	Price	Type	qty
k0001	Let us C	Y. Kanetkar	EPB	450	Prog	15
p0001	Computer Networks	B. Agarwal	FIRST PUBL	755	Comp	24
m0001	Mastering C++	K.R. Venugopal	EPB	165	Prog	60
n0002	VC++ advance	P. Purohit	TDH	250	Prog	45
k0002	Programming with Python	Sanjeev	FIRST PUBL	350	Prog	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 display the BOOK_NAME and price of the books, more than 3 copies of which have been issued.
(v) To show total cost of books of each type.
(vi) To show the details of the costliest book.

23. Write SQL commands for (i) to (vi) and write output for (vii) on the basis of PRODUCTS relation given below:

PRODUCTS TABLE

PCODE	PNAME	COMPANY	PRICE	STOCK	MANUFACTURE	WARRANTY
P001	TV	BPL	10000	200	2018-01-12	3
P002	TV	SONY	12000	150	2017-03-23	4
P003	PC	LENOVO	39000	100	2018-04-09	2
P004	PC	COMPAQ	38000	120	2019-06-20	2
P005	HANDYCAM	SONY	18000	250	2017-03-23	3

(i) To show details of all PCs with stock more than 110.
(ii) To list the company which gives warranty of more than 2 years.
(iii) To find stock value of the BPL company where stock value is the sum of the products of price and stock.
(iv) To show number of products from each company.
(v) To count the number of PRODUCTS which shall be out of warranty on 20-NOV-2020.

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

(vii) Give the output of the following statements:

(a) `SELECT COUNT(DISTINCT COMPANY) FROM PRODUCT;`

(b) `SELECT MAX(PRICE) FROM PRODUCT WHERE WARRANTY<=3;`

24. What are DDL and DML?

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

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

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

28. 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 travels	55000
6	Krishna Nagar	70	80	30	Yadav travels	80000
7	Vasundhara	100	110	20	Yadav travels	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

(i) To show all information of buses where capacity is more than the no. of students in order of Rtno.

(ii) To show Area_Covered for buses covering more than 20 km., but Charges less than 80000.

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

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

(v) Add a new record with the following data:
(11, "Motibagh",35,32,10, "Kisan tours", 35000)

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

(a) `SELECT SUM(Distance) FROM SchoolBus WHERE Transporter="Yadav travels";`

(b) `SELECT MIN(Noofstudents) FROM SchoolBus;`

(c) `SELECT AVG(Charges) FROM SchoolBus WHERE Transporter="Anand travels";`

(d) `SELECT DISTINCT Transporter FROM SchoolBus;`

29. Write SQL Commands for (i) to (v) and write the outputs for (vi) to (viii) on the basis of the following table:

Table: FURNITURE

NO	ITEM	TYPE	DATEOFSTOCK	PRICE	DISCOUNT
1	WhiteLotus	DoubleBed	2002-02-23	3000	25
2	Pinkfeathers	BabyCot	2002-01-29	7000	20
3	Dolphin	BabyCot	2002-02-19	9500	20
4	Decent	OfficeTable	2002-02-01	25000	30
5	Comfortzone	DoubleBed	2002-02-12	25000	30
6	Donald	BabyCot	2002-02-24	6500	15

(i) To list the details of furniture whose price is more than 10000.

(ii) To list the Item name and Price of furniture whose discount is between 10 and 20.

(iii) To delete the record of all items where discount is 30.

(iv) To display the price of 'Babycot'.

(v) To list item name, type and price of all items whose names start with 'D'.

(vi) `SELECT DISTINCT Type FROM Furniture;`

(vii) `SELECT MAX(Price) FROM Furniture WHERE DateofStock>'2002-02-15';`

(viii) `SELECT COUNT(*) FROM Furniture WHERE Discount<25;`

30. Write SQL Commands/output for the following on the basis of the given table GRADUATE:

Table: GRADUATE

SL.No.	NAME	STIPEND	SUBJECT	AVERAGE	RANK
1	KARAN	400	PHYSICS	68	1
2	RAJ	450	CHEMISTRY	68	1
3	DEEP	300	MATHS	62	2
4	DIVYA	350	CHEMISTRY	63	1
5	GAURAV	500	PHYSICS	70	1
6	MANAV	400	CHEMISTRY	55	2
7	VARUN	250	MATHS	64	1
8	LIZA	450	COMPUTER	68	1
9	PUJA	500	PHYSICS	62	1
10	NISHA	300	COMPUTER	57	2

(i) List the names of those students who have obtained rank 1 sorted by NAME.

(ii) Display a list of all those names whose AVERAGE is greater than 65.

(iii) Display the names of those students who have opted COMPUTER as a SUBJECT with an AVERAGE of more than 60.

(iv) List the names of all the students in alphabetical order.

(v) SELECT * FROM GRADUATE WHERE NAME LIKE "% I %";

(vi) SELECT DISTINCT RANK FROM GRADUATE;

31. (a) What is the difference between Candidate key and Alternate key?

(b) What is the degree and cardinality of a table having 10 rows and 5 columns?

(c) For the given table, do as directed:

Table: STUDENT

ColumnName	Data type	size	Constraint
ROLLNO	Integer	4	Primary Key
SNAME	Varchar	25	Not Null
GENDER	Char	1	Not Null
DOB	Date		Not Null
FEES	Integer	4	Not Null
HOBBY	Varchar	15	Null

(i) Write SQL query to create the table.

(ii) Write SQL query to increase the size of SNAME to hold 30 characters.

(iii) Write SQL query to remove the column HOBBY.

(iv) Write SQL query to insert a row in the table with any values of your choice that can be accommodated there.

32. Write SQL queries based on the following tables:

Table: PRODUCT

P_ID	ProductName	Manufacturer	Price	Discount
TP01	Talcum Powder	LAK	40	NULL
FW05	Face Wash	ABC	45	5
BS01	Bath Soap	ABC	55	NULL
SH06	Shampoo	XYZ	120	10
FW12	Face Wash	XYZ	95	NULL

Table: CLIENT

C_ID	ClientName	City	P_ID
01	Cosmetic Shop	Delhi	TP01
02	Total Health	Mumbai	FW05
03	Live Life	Delhi	BS01
04	Pretty Woman	Delhi	SH06
05	Dreams	Delhi	FW12

(i) Write SQL Query to display ProductName and Price for all products whose Price is in the range 50 to 150.

(ii) Write SQL Query to display details of products whose manufacturer is either XYZ or ABC.

(iii) Write SQL query to display ProductName, Manufacturer and Price for all products that are not giving any discount.

(iv) Write SQL query to display ProductName and price for all products whose ProductName ends with 'h'.

(v) Write SQL query to display ClientName, City, P_ID and ProductName for all clients whose city is Delhi.

(vi) Which column is used as Foreign Key and name the table where it has been used as Foreign key.

33. Answer the questions based on the table given below:

Table: HOSPITAL

SNo	Name	Age	Department	Dateofadm	Charges	Sex
1	Arpit	62	Surgery	1998-01-21	300	M
2	Zareena	22	ENT	1997-12-12	250	F
3	Kareem	32	Orthopaedic	1998-02-19	200	M
4	Arun	12	Surgery	1998-01-11	300	M
5	Zubin	30	ENT	1998-01-12	250	M
6	Ketaki	16	ENT	1998-02-24	250	F
7	Ankit	29	Cardiology	1998-02-20	800	F
8	Zareen	45	Gynaecology	1998-02-22	300	F
9	Kush	19	Cardiology	1998-01-13	800	M
10	Shilpa	23	Nuclear Medicine	1998-02-21	400	F

(a) To list the names of all the patients admitted after 15/01/98.

(b) To list the names of female patients who are in ENT department.

(c) To list the names of all patients with their date of admission in ascending order.

(d) To display Patient's Name, Charges, Age for only female patients.

(e) Find out the output of the following SQL commands:

(i) `SELECT COUNT(DISTINCT Charges) FROM HOSPITAL;`

(ii) `SELECT MIN(Age) FROM HOSPITAL WHERE Sex="F";`

34. A department store MyStore is considering to maintain their inventory using SQL to store the data. As a database administrator, Abhay has decided that: [CBSE Sample Paper 2020-21]

- Name of the database – mystore
- Name of the table – STORE

The attributes of STORE are as follows:

ItemNo – numeric

ItemName – character of size 20

Scode – numeric

Quantity – numeric

Table: STORE

ItemNo	ItemName	Scode	Quantity
2005	Sharpener Classic	23	60
2003	Ball Pen 0.25	22	50
2002	Gel Pen Premium	21	150
2006	Gel Pen Classic	21	250
2001	Eraser Small	22	220
2004	Eraser Big	22	110
2009	Ball Pen 0.5	21	180

(a) Identify the attribute best suitable to be declared as a primary key.

(b) Write the degree and cardinality of the table STORE.

(c) Insert the following data into the attributes ItemNo, ItemName and SCode respectively in the given table STORE. ItemNo = 2010, ItemName = "Note Book" and Scode = 25.

(d) Abhay wants to remove the table STORE from the database MyStore. Which command will he use from the following?

(i) `DELETE FROM STORE;`
(ii) `DROP TABLE STORE;`
(iii) `DROP DATABASE MYSTORE;`
(iv) `DELETE STORE FROM MYSTORE;`

(e) Now Abhay wants to display the structure of the table STORE, *i.e.*, name of the attributes and their respective data types that he has used in the table. Write the query to display the same.

35. Differentiate between Primary key and Unique constraints.

CASE-BASED/SOURCE-BASED INTEGRATED QUESTIONS

1. In a multiplex, movies are screened in different auditoriums. One movie can be shown in more than one auditorium. In order to maintain the record of movies, the multiplex maintains a relational database consisting of two relations, viz. MOVIE and AUDI respectively as shown below:

Movie(Movie_ID, MovieName, ReleaseDate)
Audi(AudiNo, Movie_ID, Seats, ScreenType, TicketPrice)

(a) Is it correct to assign Movie_ID as the primary key in the MOVIE relation? If not, then suggest an appropriate primary key.

Ans. Yes, Movie_ID should be the primary key in the MOVIE relation.

(b) Is it correct to assign AudiNo as the primary key in the AUDI relation? If not, then suggest an appropriate primary key.

Ans. Yes, AudiNo should be assigned as the primary key in the AUDI relation.

(c) Is there any foreign key in any of these relations?

Ans. Movie_ID is the foreign key in the relation Audi.

2. (a) Ajay wants to write an SQL query to increase the size of the 'name' column in 'Student' table to accommodate 30 characters. However, he commits an error in the SQL syntax. Rewrite the following SQL statement after removing the errors.

```
ALTER Student TABLE INCREASE Name VARCHAR(30);
```

Ans. ALTER TABLE Student MODIFY Name VARCHAR(30);

(b) Also help him to write an SQL query to create the 'Sales' table from the given data:

Table: Sales

ColumnName	Data type	Size	Constraint
Orderid	Integer	4	Primary Key
Pname	Varchar	20	Not Null
Quantity	Integer	5	Not Null
Rate	Integer	5	Not Null
Discount	Integer	3	Null

Ans.
```
CREATE TABLE Sales(
    Orderid int(4)) Primary key,
    Pname Varchar(20) Not Null,
    Quantity int(5) Not Null,
    Rate int(5) Not Null,
    Discount int(3));
```

3. A shop called Trends Garments that sells school uniforms maintains a database SCHOOL_UNIFORM as shown below. It consists of two relations—UNIFORM and PRICE. They made Uniform_Code as the primary key for UNIFORM relation. Further, they used Uniform Code as the primary key in the PRICE relation.

UNIFORM

Uniform_Code	Uniform_Name	Uniform_Color
1	Shirt	White
2	Trouser	Grey
3	Skirt	Blue
4	Tie	Grey
5	Socks	Blue and Grey Checks
6	Belt	Blue
7	Blazer	Maroon

PRICE

Uniform_Code	Size	Price
1	XL	480
1	L	450
2	L	520
2	M	490
2	XXL	550
3	S	540
3	M	560
3	L	580
4	S	100
4	M	120
5	M	120
5	S	100
6	L	100
6	XL	120
7	L	1000
7	XL	1100

(a) The PRICE relation has an attribute named Price. In order to avoid confusion, write SQL query to change the name of the field Price to Cost.

Ans.
```
ALTER TABLE PRICE
CHANGE Price Cost INTEGER;
```

(b) M/S Trends Garments also keeps BLAZER of MAROON colour in medium size, priced at ₹ 900 each. Insert this record in PRICE table.

Ans.
```
INSERT INTO PRICE VALUES(7,'M',900);
```

(c) ALTER table to add the constraint that price of an item is always greater than zero.

Ans.
```
ALTER TABLE PRICE
MODIFY COST CHECK COST>0;
```