**Task 1**

**1:**

A collection of data that is stored, managed, and easily accessible. Data requirements help in storing, managing, and retrieving data in an efficient manner

**2:**

A programming language for information technology in relational databases. This is why SQL users perform a variety of operations on data

**Task 2:**

**1:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature** | |  | | --- | | **Relational Databases** |  |  | | --- | |  | | | **Non-relational Databases (NoSQL)** | | --- |  |  | | --- | |  | |
| |  | | --- | | **Data Structure** |  |  | | --- | |  | | |  | | --- | | Tables with predefined schema |  |  | | --- | |  | | |  | | --- | | Flexible schema, often schema-less or dynamic |  |  | | --- | |  | |
| |  | | --- | | **Query Language** |  |  | | --- | |  | | |  | | --- | | SQL (Structured Query Language) |  |  | | --- | |  | | |  | | --- | | Varies by database (e.g., MongoDB uses JSON-like queries) |  |  | | --- | |  | |
| **Complexity** | |  | | --- | | Strict schema with predefined structure |  |  | | --- | |  | | |  | | --- | | Dynamic schema, easy to change data structure |  |  | | --- | |  | |

**2:**

MySQL is a widely-used open-source relational database management system known for its scalability, reliability, and speed. It employs SQL for handling data operations and provides strong security features and transaction support. MySQL offers flexibility through various storage engines tailored for different workload types. It finds application in diverse scenarios including web applications like WordPress, e-commerce platforms, real-time transaction systems, and data warehousing for analytics. Its ability to run on multiple platforms and its large community support make it a preferred database solution across industries seeking efficient management of structured data.

**Task 3 :**

MySQL offers several numeric data types to accommodate different ranges and precision requirements:

1. **INTEGER (or INT)**: Used for whole numbers, it can store values from -2,147,483,648 to 2,147,483,647 (signed) or 0 to 4,294,967,295 (unsigned). Example: age INT (Signed INT to store age).
2. **DECIMAL**: Ideal for exact numeric values where precision is crucial, such as financial data. It allows for precise storage of numbers up to 65 digits, with specified precision. Example: price DECIMAL(10, 2) (To store prices with up to 10 digits total, with 2 digits after the decimal point).
3. **FLOAT**: Used for approximate numeric values, suitable for scientific calculations or where precision beyond integers is required. Example: temperature FLOAT (To store temperature values).
4. **BIGINT**: Supports very large integers, ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (signed) or 0 to 18,446,744,073,709,551,615 (unsigned). Example: population BIGINT (To store population numbers).

These data types provide flexibility in storing various numeric values while ensuring efficient storage and retrieval operations in MySQL databases.

Task4

1. **CHAR (size):**
2. A fixed-length string that can contain letters, numbers, and special characters. The size parameter specifies the column length in characters (from 0 to 255). The default size is 1.
3. **VARCHAR (size):**
4. A variable-length string that can also contain letters, numbers, and special characters. The size parameter specifies the maximum column length in characters (from 0 to 65535).
5. **BINARY (size):**
6. Similar to CHAR, but stores binary byte strings. The size parameter specifies the column length in bytes. Default size is 1.
7. **VARBINARY (size):**
8. Similar to VARCHAR, but for binary byte strings. The size parameter specifies the maximum column length in bytes.
9. **TINYBLOB:**
10. For Binary Large Objects (BLOBs) with a maximum length of 255 bytes.
11. **TINYTEXT:**
12. Holds a string with a maximum length of 255 characters.
13. **TEXT (size):**
14. Holds a string with a maximum length of 65,535 bytes.
15. **BLOB (size):**
16. For BLOBs, holding up to 65,535 bytes of data.
17. **MEDIUMTEXT:**
18. Holds a string with a maximum length of 16,777,215 characters.
19. **MEDIUMBLOB:**
20. For BLOBs, holding up to 16,777,215 bytes of data.
21. **LONGTEXT:**
22. Holds a string with a maximum length of 4,294,967,295 characters.
23. **LONGBLOB:**
24. For BLOBs, holding up to 4,294,967,295 bytes of data.
25. **ENUM (val1, val2, val3, …):**
26. A string object with a single value chosen from a list of possible values. You can list up to 65535 values in an ENUM list.
27. **SET (val1, val2, val3, …):**
28. A string object that can have 0 or more values from a list. You can list up to 64 values in a SET list.

Task 5

date

* Description: Stores date values in the format 'YYYY-MM-DD'. It does not store time information.
* Example: DATE (Stores dates like '2020-08-13').

datetime:

* Description: Stores date and time values in the format 'YYYY-MM-DD HH:MM

'.

* Example: DATETIME (Stores date and time like '2020-08-13 12:30:45').

timestamp:

* Description: Stores date and time values in the format 'YYYY-MM-DD HH:MM

'. It has automatic updating behavior and adjusts to the current time zone settings of the server.

* Example: TIMESTAMP (Automatically updates with the current timestamp).

time

* Description: Stores time values in the format 'HH:MM

'.

* Example: TIME (Stores time like '12:30:00').

year

* Description: Stores only the year part of a date.
* Example: YEAR (Stores years like '2020')

Task 6

A primary key in a relational database is a unique identifier assigned to each row within a table. It serves two fundamental purposes:

Uniqueness: A primary key ensures that each record in a table can be uniquely identified. No two rows can have the same primary key value, which helps maintain data integrity and prevents duplicate entries.

Indexing and Performance: Primary keys are often indexed by the database management system (DBMS), allowing for fast retrieval and efficient querying of data. This indexing enhances database performance, especially when dealing with large datasets or complex queries.

Overall, a primary key plays a crucial role in structuring databases by ensuring data uniqueness and enabling efficient data retrieval. It serves as a cornerstone for relational databases' relational model, supporting relationships between tables and facilitating the maintenance of data consistency and integrity.

Task7

**NULL Constraint**:

* **Description**: Allows a column to store NULL values, indicating that the data for that column is missing, unknown, or not applicable.

**middle\_name VARCHAR(50) NULL**

to be empty for some records.

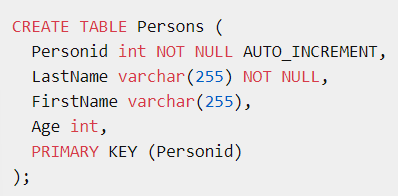
not **NULL Constraint**:

* **Description**: Requires a column to have a value. It ensures that every row must contain data for that column, and NULL values are not allowed.

last\_name VARCHAR(50) NOT NULL

task8

**This field is often used as a PRIMARY KEY, where a unique value must be provided for each record added.**



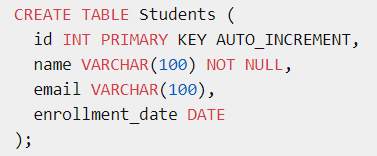
Task9



Task10

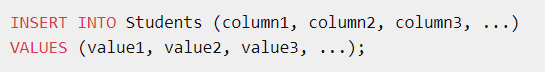


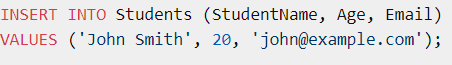
Task11



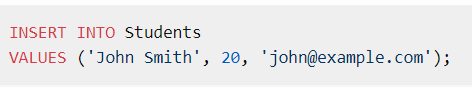
Task12

1

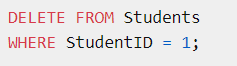




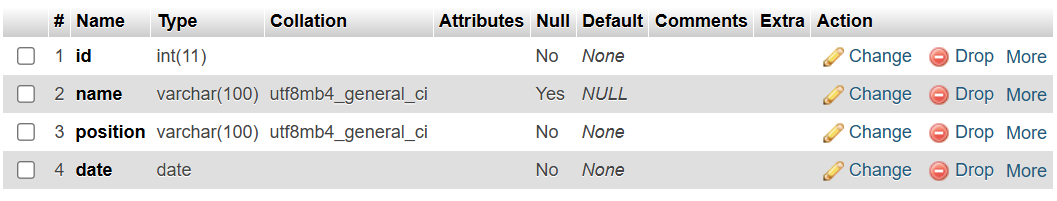
2



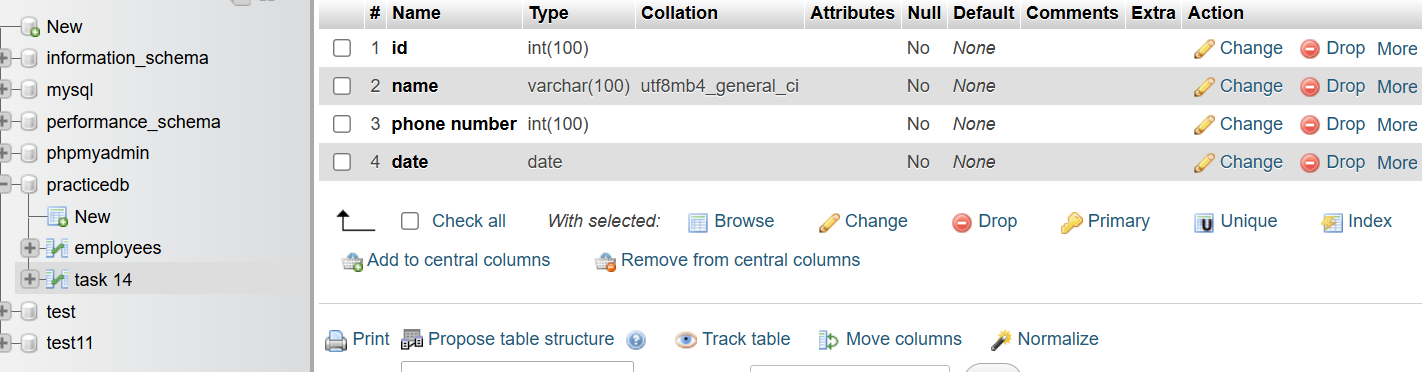
3



Task13



Task 14



Task 15