Experiment No.3
Create a database using Data Definition Language(DDL) and apply integrity constraints for the specified system
Date of Performance:
Date of Submission:

Aim:- Write a query to create tables for each relation in the relational schema of experiment no.2. Apply drop and alter commands on those tables.

Objective:- To learn commands of Data Definition Language(DDL) to create and define databases, and also learn to apply integrity constraints for the specified system.

Theory:

DDL Commands & Syntax:-

Data Definition Language (DDL) is a subset of SQL and a part of DBMS(Database Management System). DDL consist of Commands to commands like CREATE, ALTER, TRUNCATE and DROP. These commands are used to create or modify the tables in SQL. DDL Commands:

1. Create
2. Alter
3. truncate
4. drop
5. Rename

CREATE:

This command is used to create a new table in SQL. The user must give information like table name, column names, and their data types.

Syntax –CREATE TABLE table_name

(

column_1 datatype,

column_2 datatype,

column_3 datatype,

....

);

ALTER :

This command is used to add, delete or change columns in the existing table. The user needs to know the existing table name and can add, delete, or modify tasks easily.

Syntax –

ALTER TABLE table_name

ADD column_name datatype;

TRUNCATE :

This command is used to remove all rows from the table, but the structure of the table still exists.

Syntax –

TRUNCATE TABLE table_name;

DROP :

This command is used to remove an existing table along with its structure from the Database.

Syntax –

DROP TABLE table_name;

RENAME :

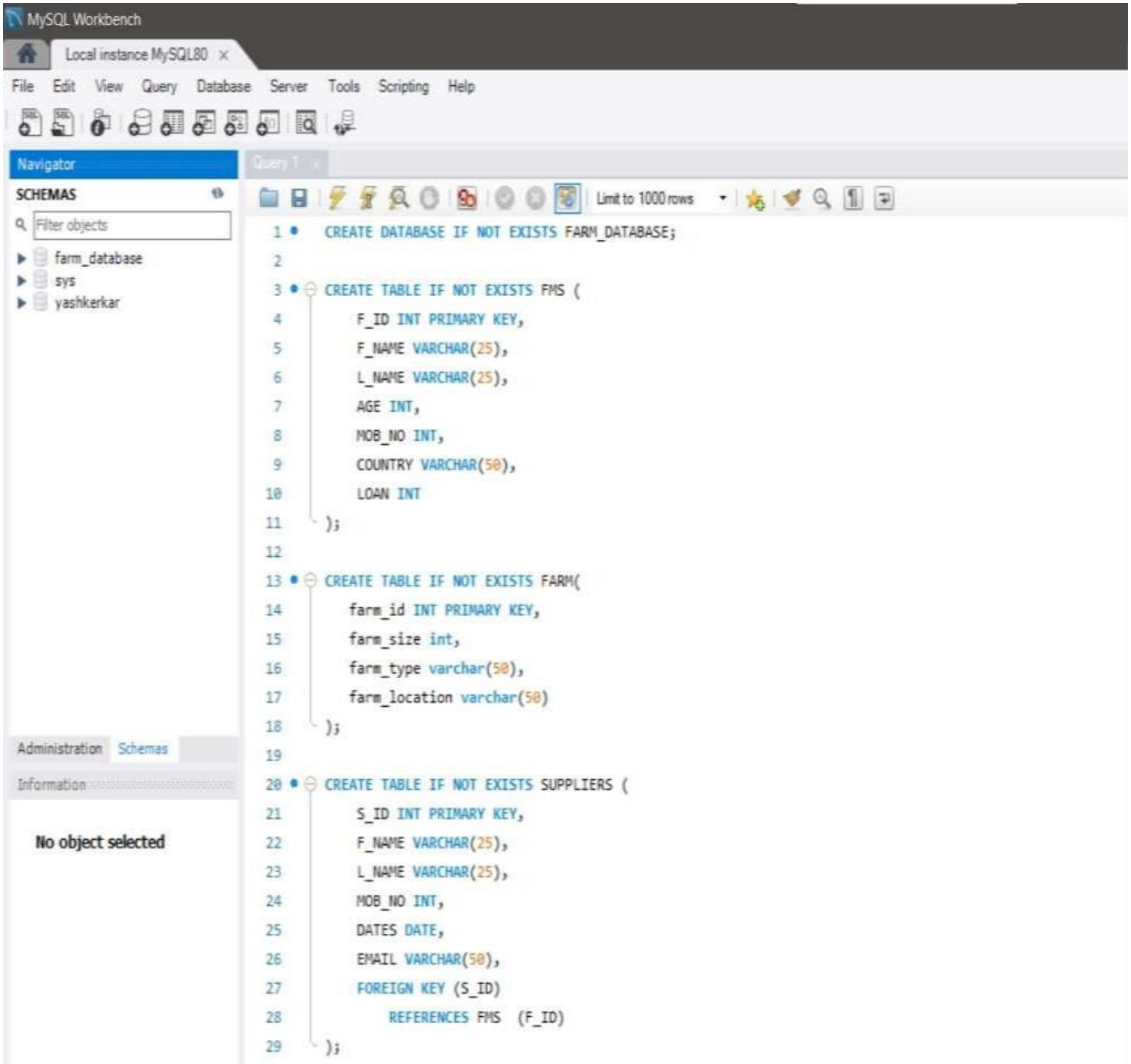
It is possible to change name of table with or without data in it using simple RENAME command. We can rename any table object at any point of time.

Syntax –

RENAME TABLE <Table Name> To <New_Table_Name>;

Implementation:

Program:-



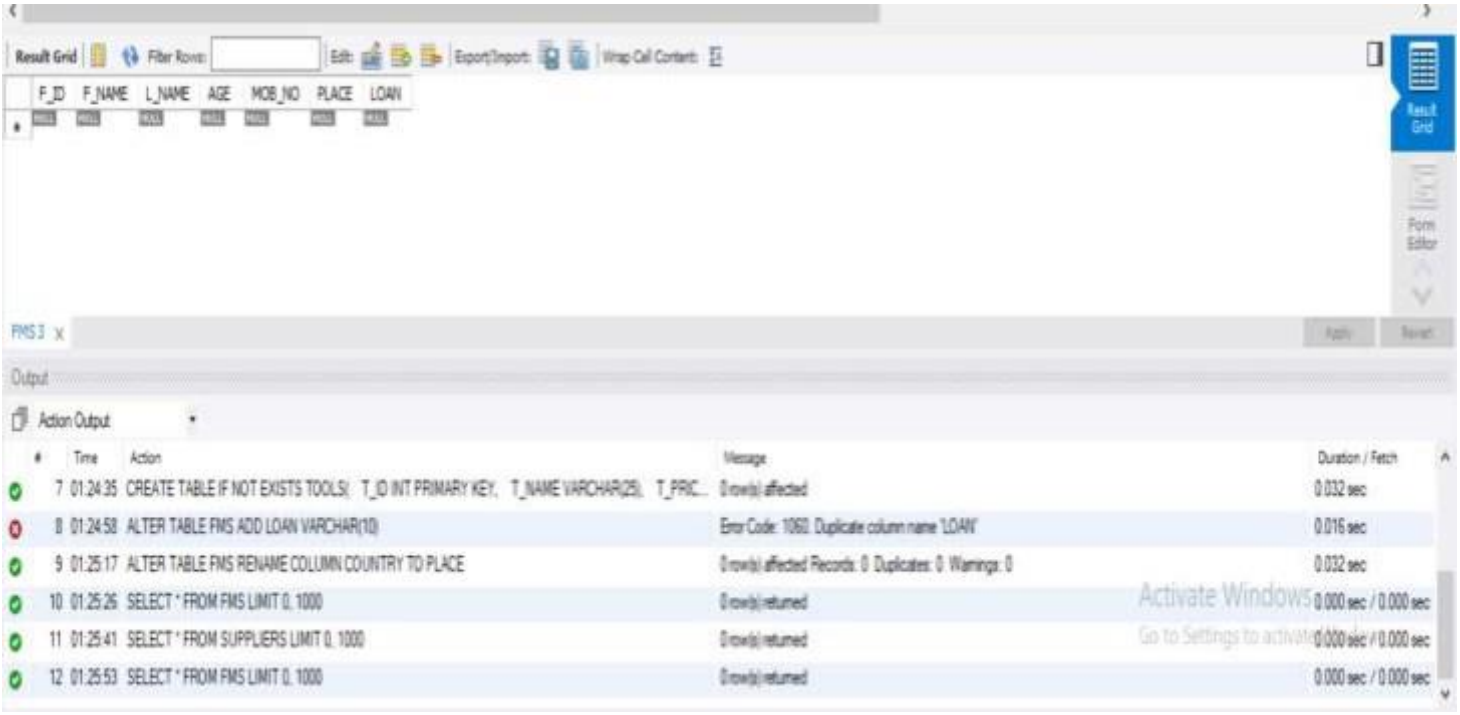
Query 1

Limit to 1000 rows

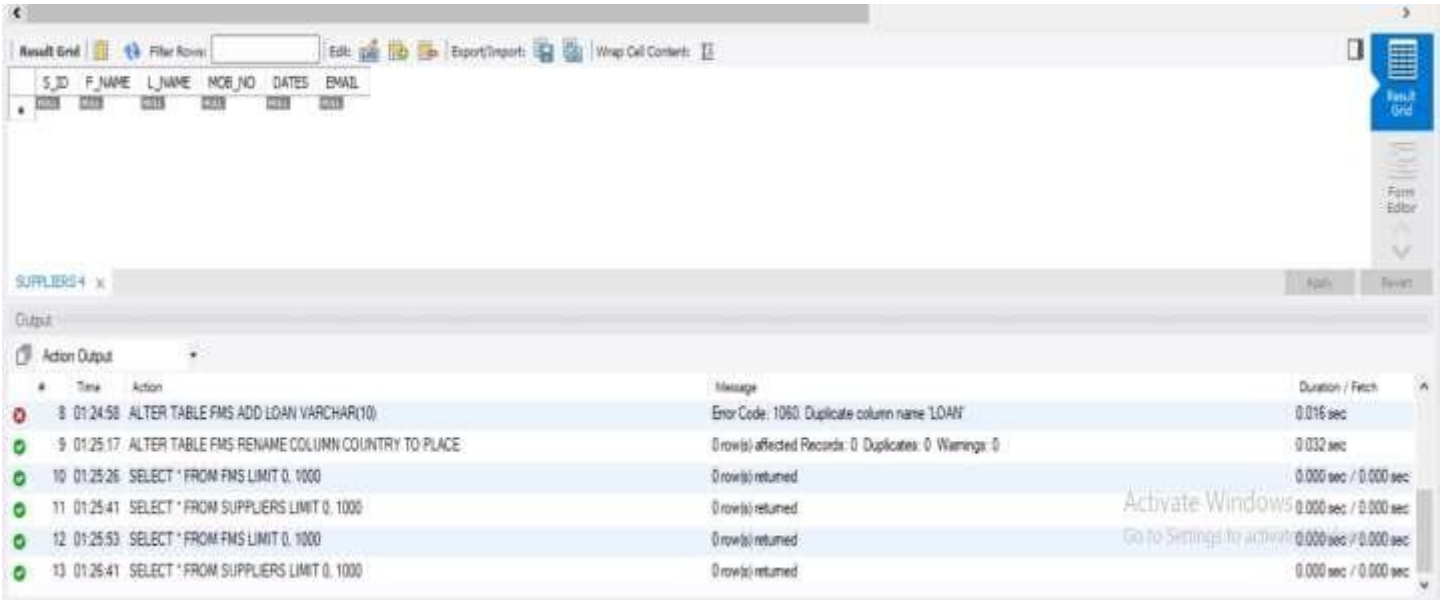
```
29      );
30
31  •   CREATE TABLE IF NOT EXISTS TOOLS(
32          T_ID INT PRIMARY KEY,
33          T_NAME VARCHAR(25),
34          T_PRICE INT,
35          T_DESC VARCHAR(200)
36      );
37
38  •   ALTER TABLE FMS ADD LOAN VARCHAR(10);
39  •   ALTER TABLE FMS DROP LOAN;
40
41  •   ALTER TABLE FMS
42      RENAME COLUMN COUNTRY TO PLACE;
43
44  •   TRUNCATE TABLE FMS;
45  •   TRUNCATE TABLE SUPPLIERS;
46  •   TRUNCATE TABLE FARM;
47  •   TRUNCATE TABLE TOOLS;
48
49  •   DROP TABLE FMS;
50  •   DROP TABLE SUPPLIERS;
51  •   DROP TABLE FARM;
52  •   DROP TABLE TOOLS;
53
54  •   SELECT * FROM FMS;
55  •   SELECT * FROM SUPPLIERS;
56  •   SELECT * FROM FARM;
57  •   SELECT * FROM TOOLS;
```

Output:

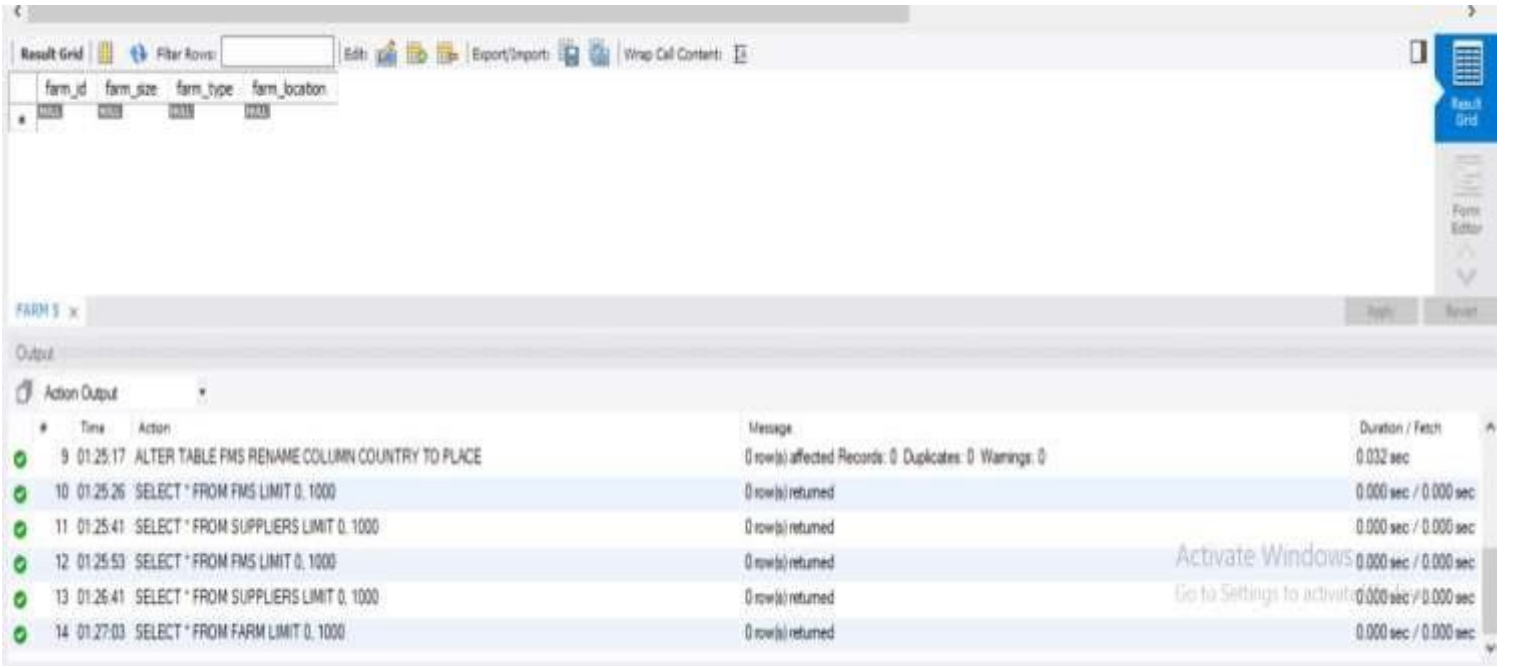
Create Farmers Table:



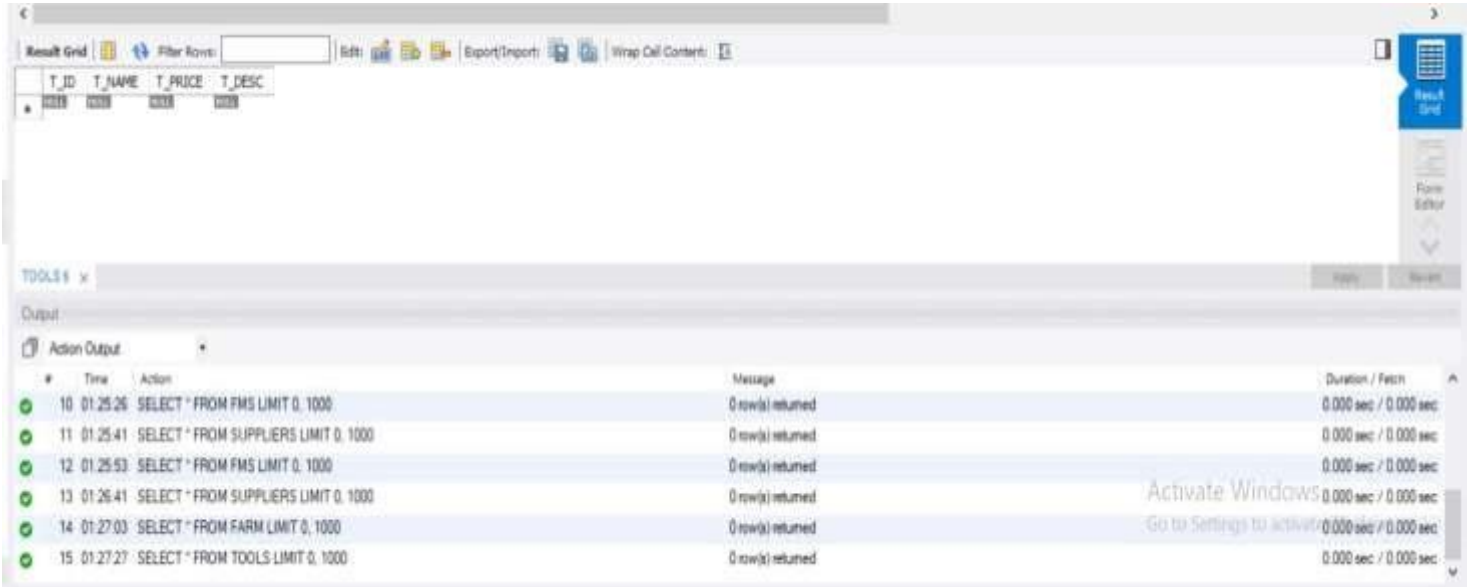
Create Suppliers Table:



Create Farm Table



Create Tools Table



Conclusion:

The completion of the practical exercise involving Data Definition Language (DDL) commands on the Farmers Management System offers valuable insights into the foundational aspects of database management. By applying various DDL commands, including CREATE TABLE, ALTER TABLE, and DROP TABLE, we gained hands-on experience in defining, modifying, and deleting the structure of the database schema.

1. Explain the concept of constraints in DDL. How are constraints used to enforce data integrity?

Ans. Constraints in Data Definition Language (DDL) are rules or conditions applied to the structure of a database schema to enforce data integrity and maintain consistency. They define limitations and restrictions on the data that can be stored in the database, ensuring that only valid, accurate, and meaningful data is allowed. Constraints play a crucial role in database design by preventing data inconsistencies, errors, and anomalies.

Here's how constraints are used to enforce data integrity in a database:

1. **Primary Key Constraint**: A primary key constraint ensures that each row in a table is uniquely identifiable by a primary key attribute or combination of attributes. It prohibits duplicate and null values in the primary key column(s), thereby enforcing entity integrity and ensuring that each record is uniquely identifiable.

2. **Unique Constraint**: The unique constraint ensures that the values in one or more columns of a table are unique across all rows. Unlike primary keys, unique constraints allow null values, but if a column is marked as unique, only one row may contain a null value in that column. This constraint helps maintain data integrity by preventing the insertion of duplicate values in specified columns.

3. **Foreign Key Constraint**: Foreign key constraints establish relationships between tables by enforcing referential integrity. A foreign key in one table references the primary key in another table, ensuring that every foreign key value must match a primary key value.

4. **Check Constraint**: Check constraints define conditions that must be true for every row in a table. They allow you to specify rules that restrict the values allowed in certain columns. For example, a check constraint can ensure that values in a "age" column are greater than zero and less than 120. Check constraints help enforce domain integrity by validating the data against predefined conditions.

5. **Not Null Constraint**: The not null constraint ensures that a column cannot contain null values. It requires that every row in the table must have a value for the specified column, preventing the insertion of null values. This constraint helps maintain data integrity by ensuring that essential attributes are always populated with valid values.

6. **Default Constraint**: Default constraints specify a default value for a column when no value is explicitly provided during insertion. If a column with a default constraint is not specified in an INSERT statement, the default value will be used. Default constraints help ensure consistency and streamline data entry by providing default values for optional attributes.

By enforcing these constraints, database management systems (DBMS) ensure that the data stored in the database meets predefined rules and criteria, thereby safeguarding data

integrity and reliability. Constraints help maintain the accuracy, consistency, and validity of data, making them an essential component of database design and management.

2. What is the significance of data types in DDL? Provide examples of commonly used data types in DDL.

Ans. Data types in Data Definition Language (DDL) specify the type of data that can be stored in a column of a table in a relational database. They define the format, size, and range of values that a particular attribute can hold. Data types are essential for ensuring data integrity, optimizing storage, and facilitating efficient data retrieval and manipulation operations.

The significance of data types in DDL can be summarized as follows:

Data Integrity: Data types enforce constraints on the values that can be stored in a column, helping to maintain data integrity. By specifying the appropriate data type for each attribute, DDL ensures that only valid and meaningful data is stored in the database.
Storage Optimization: Different data types require different amounts of storage space. Choosing the most appropriate data type for each attribute can optimize storage utilization and reduce storage requirements. For example, using a smaller data type like INTEGER instead of BIGINT for a column that only needs to store small integers can conserve storage space.
Data Retrieval and Manipulation Efficiency: Data types affect the efficiency of data retrieval and manipulation operations. Using appropriate data types can improve query performance and reduce processing overhead. For example, indexing columns with suitable data types can speed up search and retrieval operations.
Data Validation: Data types help validate the format and range of values entered into a column. They prevent the insertion of invalid or incompatible data, thereby enhancing data quality and consistency. For example, using a DATE data type for a column ensures that only valid dates can be stored in that column.
Examples of commonly used data types in DDL include:

INTEGER: Stores whole numbers (e.g., 1, 10, -5).
FLOAT: Stores floating-point numbers (e.g., 3.14, -0.001).
CHAR(n): Stores fixed-length character strings of length n (e.g., 'hello').
VARCHAR(n): Stores variable-length character strings of maximum length n (e.g., 'world').
DATE: Stores date values in YYYY-MM-DD format (e.g., '2022-04-19').
TIME: Stores time values in HH:MM:SS format (e.g., '14:30:00').
BOOLEAN: Stores true/false or 1/0 values.
NUMERIC(p, s): Stores fixed-point numbers with precision p and scale s (e.g., NUMERIC(10, 2) for monetary values).