Introtallent

Training | Analytics | Consulting

Helping People Decode Analytics for Business

SQL

Data Definition Language (DDL)

Create Database Statement

```
/*Show the list of existing databases*/
SHOW databases;
```

The **CREATE DATABASE** statement is used to create a new SQL database.

```
/*Create a database by name training*/
CREATE DATABASE training;
```

The **DROP DATABASE** statement is used to drop an existing SQL database.

```
/*DROP (Delete) training database*/
DROP DATABASE training;

DROP DATABASE IF EXISTS training;
```

DROP DATABASE IF EXISTS training; will delete the database if it exists else ignore.

DROP DATABASE training; will return an error if the database doesn't exist.

CREATE & DROP Table Statement

```
/*Use training database in following commands*/
USE training;
/*Show the list of existing tables in training database*/
SHOW tables;
```

The **CREATE TABLE** statement in SQL is used to create a table.

```
/*Create Student table in training database*/
CREATE TABLE training.Student
(
    Student_ID int Primary Key,
    LastName varchar(255) Not Null,
    FirstName varchar(255) Not Null,
    Address varchar(255),
    City varchar(255)
);
```

The **DROP TABLE** statement is used to drop an existing table in a database.

```
/*DROP Student table from training database*/
DROP TABLE training.Student ;
DROP DATABASE IF EXISTS training.Student;
```

ALTER Table Statement

The ALTER table statement is used:

- to add a new column to the table structure
- drop an existing column from the table structure
- to modify columns in an existing table
- to add and drop various constraints on an existing table.

You may need it after table creation is done and later you want to add or delete a column.

```
/*Create Student table in training database*/
CREATE TABLE training.Student
(
    Student_ID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

```
/*Use DESC to check the table structure*/
DESC training.student;

/*Add a new column Age, after the column FirstName in the table above*/
ALTER table training.Student
ADD column Age INT after FirstName;

/*Delete the column Age*/
ALTER table training.Student
DROP column Age;
```

Auto_Increment Field

- If you want to generate a number automatically when a new record is inserted into a table then you can use AUTO_INCREMENT.
- *Often this is the primary key field that we would like to be created automatically every time a new record is inserted.*

```
/*Create Student table in training database*/
CREATE TABLE training.Student
(
Student_ID int AUTO_INCREMENT Primary key,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255) );
```

***NOTE**: There can be only one auto column and it must be defined as a key*

- MySQL uses the AUTO_INCREMENT keyword to perform an auto-increment feature.
- By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record.
- To let the AUTO_INCREMENT sequence start with another value, use the following SQL statement:

```
/*Start the auto increment from 100*/
ALTER TABLE training.student
AUTO_INCREMENT=100;
```

Temporary Table

- In MySQL, a temporary table is a special type of table that allows you to store a temporary result set, which you can reuse several times in a single session.
- A temporary table is very handy when it is impossible or expensive to query data that requires a single SELECT statement. In this case, you can use a temporary table to store the immediate result and use another query to process it.
- *MySQL removes the temporary table automatically when the session ends or the connection is terminated. Of course, you can use the DROP TABLE statement to remove a temporary table explicitly when you are no longer use it.*
- A temporary table can have the same name as a normal table in a database but in that case the actual table will become inaccessible. For example, if you create a temporary table named employees in the sample database, the existing employees table becomes inaccessible.

Important: Even though a temporary table can have the same name as a permanent table, it is not recommended. Because this may lead to a confusion and potentially cause an unexpected data loss.

```
/*Here is how you can create a temporary table*/
CREATE TEMPORARY  TABLE
training.temp_Student
(
Student_ID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);

/*Dropping a  temporary table*/
DROP TEMPORARY TABLE training.temp_Student;
```

VIEWS

VIEWS are virtual tables. By virtual, we mean, the tables do not store any data of their own but display data stored in other tables.

```
/*Create a VIEW by selecting few columns from country table*/
Create VIEW training.v_country
As
Select continent, name, population
from world.country;
```

- *Views are virtual tables; they do not contain the data that is returned. The data is stored in the tables referenced in the SELECT statement.*
- *Views improve security of the database by showing only intended data to authorized users.*
- *Views make life easy as you do not have write complex queries time and again.*

- *It's possible to use INSERT, UPDATE and DELETE on a VIEW. These operations will change the underlying tables of the VIEW. The only consideration is that VIEW should contain all NOT NULL columns of the tables it references. Ideally, you should not use VIEWS for updating.*

Dropping VIEWS

The DROP command can be used to delete a view from the database that is no longer required. The basic syntax to drop a view is as follows.

```
/*Delete VIEWS using DROP command*/
DROP VIEW training.v_student;
```

Creating table using SELECT statement

```
/*Create a table by selecting data from world.country*/

create table training.abc
(
Select * from world.country
);

/*Create a temporary table by selecting data from world.country*/

create TEMPORARY table temp_training.abc
(
Select name, continent, population
from world.country
);
```

You can also create table by combining data from multiple tables using JOINS. We will talk about JOINS in upcoming sessions.

Index Statement

- Indexes are used to find rows with specific column values quickly. Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows. The larger the table, the more this costs. If the table has an index for the columns in question, MySQL can quickly determine the position to seek to in the middle of the data file without having to look at all the data. This is much faster than reading every row sequentially.
- *In simple terms, Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.*

NOTE: Updating a table with indexes takes more time than updating a table without (because the indexes also need an update). So, only create indexes on columns that will be frequently searched against.

```
/*Create an index on name column in country table.
This may create duplicate indexes*/
Create INDEX idx_firstIndex on
world.country(name);

/*Create a unique index on name column in country
table. This will always create a unique index*/
Create Unique INDEX idx_firstIndex on
world.country(name);

/*Create an index on name column in country table*/
Create INDEX idx_firstIndex on
world.country(name);

/*Create an index multiple column combination*/
Create INDEX idx_firstIndex on
world.country(name, continent);

/*Drop the index */
ALTER TABLE world.country
DROP INDEX idx_firstIndex;
```

We would love to hear back!

80737 99421
861 856 9998

info@introtallent.com
www.introtallent.com

Office Address:
Introtallent Pvt Ltd.
#12, Anu Arcade, 3rd Floor
CMH Road, Indiranagar,
Bangalore – 560038