

Lab 08 Manual
Database Systems.



Objective:

To explore concepts i.e., DDL, DML, DCL and DQL.

Scope:

The student shall know the following:

- Workaround SQL Server.
- SQL Practice.
- Hands-on experience on the above-mentioned concepts.

Discussion:

SQL Language is made up of various types of SQL Commands. The SQL language commands are divided into four main sections:

- **DML** (Data Manipulation Language) Commands:
 - DML commands are used to insert, update and delete data in the database. As the name suggests, they are used to manipulate the data in database.
 - The DML commands are INSERT, UPDATE and DELETE.
- **DDL** (Data Definition Language) Commands:
 - DDL commands are used to manipulate the database itself. They deal with creation of database, altering it's structure and deleting the database.
 - The DDL Commands are CREATE, ALTER and DROP.
- **DCL** (Data Control Language) Commands:
 - DCL commands are used to control the access to the database. They are used to give privilege to the users and take away privilege from the users.
 - The DCL Commands are GRANT and REVOKE.
- **DQL** (Data Query Language) Command:
 - Data is accessed from the database by firing queries. DQL command allows us to fire queries to the database and get the data from the database.
 - The DQL command is SELECT.

1. Data Definition Language (DDL):

The various DDL commands are **CREATE**, **ALTER** and **DROP**. Let us learn about each DDL Command one by one with examples.

Using DDL statements it is possible to:

- Define and create a new table
- Remove a table that is no longer needed
- Change the definition of an existing table
- Define a virtual table (view) of data
- Establish security controls for a database
- Build an index to make table access faster
- Control the physical storage of data by the DBMS

CREATE Statement:

The CREATE statement in SQL is used to create the database and the database objects like Tables, Views, Sequences, etc. Let us see how a database and table are created in SQL. Creating a database in SQL:

Syntax:

```
CREATE DATABASE DatabaseName;
```

Example:

```
CREATE DATABASE Demos;
```

On executing this query, a database named “Demos” is created.

CREATE TABLE in SQL:

Database is a collection of related tables and other objects. This implies that a table is a part of the database. Now, we have created the "Demos" database. Now, we need to create a “Students” table in this database.

By default, the master database is selected. Select Demos database from the dropdown list or alternatively you can use the following command:

```
USE Demos;
```

This command will instruct the SQL Server that you want to use the database **Demos**. Then we can create a table within **Demos** database.

Syntax:

```
CREATE TABLE TableName(  
    columnName1 datatype,  
    columnName2 datatype,  
    .  
    .  
    columnNameN datatype);
```

Here, datatype is the type of data which you need to store in that column. For numeric values, we have "**numeric**" datatype, for strings we have "**varchar**" and so on. We will study about the datatypes in the upcoming articles.

Example:

```
CREATE TABLE Students(  
    id numeric(3),  
    name varchar(50)  
);
```

This will create a table named **Students** with two columns, one is "**id**" which can contain a numeric value up to three digits and second "**name**" which is varchar(50), which means that it can contain 50 characters.

Now, we have created a table with two columns. And we need to add two more columns "city" and "marks" to the table Students. Then, how can we go do this?

This can be done using the **ALTER** Command of SQL.

ALTER Statement:

The Alter Table command helps us to modify the structure of the table. If we need to add, delete or modify the columns in our existing table then we use the Alter Table Statement.

Syntax:

To add column to the existing table:

```
ALTER TABLE ExistingTableName  
ADD columnName datatype;
```

To remove column from the existing table:

```
ALTER TABLE ExistingTableName  
DROP columnName;
```

Lastly, to modify the datatype of a column in our existing table:

ALTER TABLE ExistingTableName

ALTER COLUMN columnName newDataType;

Example:

Let's add the **city** and **marks** columns to our table **students**.

```
ALTER TABLE students  
ADD city varchar(25);
```

```
ALTER TABLE students  
ADD marks numeric(3);
```

DROP Statement:

As the name suggests, the DROP Statement is used to delete a table or a database.

Syntax:

To delete a table:

DROP TABLE TableName;

Similarity, to delete a database;

DROP DATABASE databaseName;

Example:

```
DROP TABLE Students;
```

This will delete the **Students** table which we have created.

There may be a situation when we just want to empty the table and not to delete the table itself.

In such situations, we use the **TRUNCATE** table statement of SQL.

Suppose we have data of 50 old students in our table. But we don't need that data anymore.

But we need the table as we need to keep a record of our students. In such situations, we will use the truncate table statement so that only the data gets deleted and the table is as it is.

Example:

```
TRUNCATE TABLE Students;
```

That's all about various DDL Statements.

2. Data Manipulation Language (DML):

The various DML Commands in SQL are **INSERT**, **UPDATE** and **DELETE**.

INSERT Statement:

Simply, the insert statement is used to insert data into the table.

Syntax:

```
INSERT INTO tableName (  
column1,  
column2,  
column3,  
.  
.  
.  
columnN) VALUES (  
column1Value,  
column2Value,  
column3Value,  
.  
.  
.  
columnNValue  
);
```

Similarly,

```
INSERT INTO tableName VALUES (  
column1Value,  
column2Value,  
column3Value,
```

.
.
.

columnNValue

);

Example:

INSERT INTO Students (
id, name, city, marks) VALUES (
1,
'Sami Ullah',
'Peshawar',
584
);

INSERT INTO Students VALUES (
2,
'Mutharib Ayub',
'Rawalpindi',
574
);

UPDATE Statement:

There may be situations when we need to modify the data that we have inserted in the table.
This can be achieved with the help of the UPDATE Statement in SQL. UPDATE
Statement allows us to modify/change the data in our tables.

Syntax:

UPDATE tableName

SET column1 = column1Value, column2 = column2Value,..... column = columnNValue

WHERE condition;

Examples:

UPDATE Students
SET name = 'Pir Sami Ullah Shah'
WHERE id = 1;

UPDATE Students
SET name = 'Pir Sami Ullah Shah', city = 'Karachi'
WHERE id = 1;

DELETE Statement:

As the name suggests, the Delete Statement is used to delete records/rows from the table.

Syntax:

DELETE from tableName

WHERE condition;

Examples:

Suppose, we need to delete the record of the student having id = 1. This can be done very easily by writing the following query.

```
DELETE FROM Students  
WHERE id = 1;
```

It is important to specify the WHERE condition along with the delete statement. If we do not specify the WHERE Condition, all the rows will be deleted.

3. SQL Server Constraints:

Constraints in SQL Server are rules and restrictions applied on a column or a table such that unwanted data can't be inserted into tables. This ensures the accuracy and reliability of the data in the database. We can create constraints on single or multiple columns of any table. Constraints maintain the data integrity and accuracy in the table. Constraints can be classified into the following two types.

1. Column Type Constraints:

Definitions of these types of constraints is given when the table is created.

Syntax:

```
CREATE TABLE tableName (  
    id int NOT NULL,  
    salary int CHECK(salary > 500)  
);
```

2. Table Type Constraints:

Definitions of these types of constraints is given after the creation of the table using the Alter Command.

Syntax:

ALTER TABLE tableName

ALTER COLUMN columnName columnDataType CHECK(columnName > 600);

Moreover, SQL Server contains the following 6 types of constraints:

1. Not Null Constraint
2. Check Constraint
3. Default Constraint
4. Unique Constraint
5. Primary Constraint
6. Foreign Constraint

Let us understand each constraint briefly.

NOT NULL CONSTRAINT:

A Not null constraint restricts the insertion of null values into a column. If we are using a Not Null Constraint for a column then we cannot ignore the value of this column during an insert of data into the table.

Column Level:**Syntax:**

```
CREATE TABLE tableName (  
    columnName datatype CONSTRAINT constraintName NOT NULL  
);
```

Example:

```
CREATE TABLE students (  
    id int CONSTRAINT not_null_constraint NOT NULL  
);
```

Table Level:**Syntax:**

ALTER TABLE tableName

ALTER COLUMN columnName columnDataType NOT NULL;

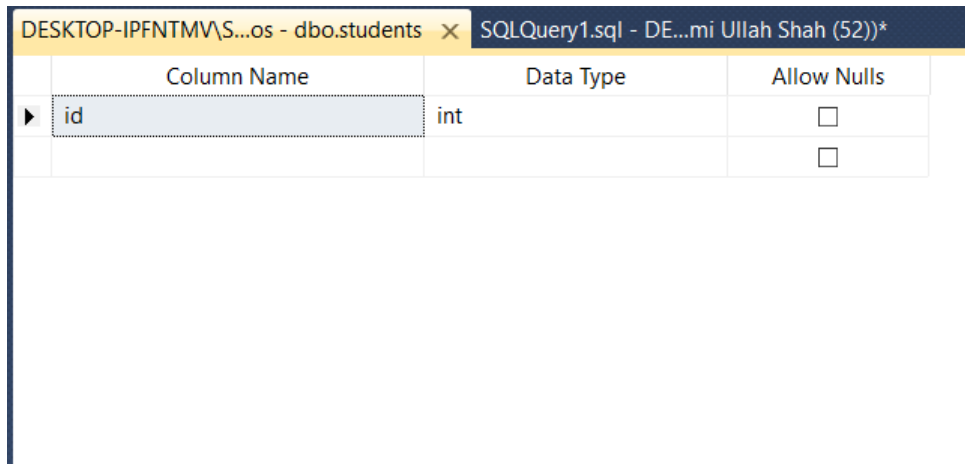
Example:

```
ALTER TABLE students  
ALTER COLUMN id int NOT NULL;
```

Without SQL Command:

We can also create a Not Null constraint in Microsoft SQL Server without execution of a SQL query.

First right-click on the table and select and click on the design option. Now check all the columns in the “Allow Nulls” option that should have a Null Value.



DESKTOP-IPFNTMV\S...os - dbo.students X SQLQuery1.sql - DE...mi Ullah Shah (52))*			
	Column Name	Data Type	Allow Nulls
▶	id	int	<input type="checkbox"/>
			<input type="checkbox"/>

CHECK Constraint:

A CHECK constraint checks for a specific condition before inserting data into a table. If the data passes all the CHECK constraints, then the data will be inserted into the table otherwise the data for insertion will be discarded. The CHECK constraint ensures that all values in a column satisfies certain conditions.

Syntax:

Column Level:

```
CREATE TABLE tableName (  
  
    columnName datatype CONSTRAINT constraintName CHECK(condition)  
  
);
```

Table Level:

```
ALTER TABLE tableName
```

```
ADD CONSTRAINT constraintName CHECK(condition);
```

DEFAULT Constraint Syntax:

Specifies a default value for when a value is not specified for this column. If in an insertion query any value is not specified for this column then the default value will be inserted into the column.

Syntax:

Column Level:

```
CREATE TABLE tableName (  
  
    columnName datatype CONSTRAINT constraintName DEFAULT(value)  
  
);
```

Table Level:

```
ALTER TABLE tableName  
  
ADD CONSTRAINT constraintName DEFAULT(value) FOR [columnName];
```

UNIQUE Constraint Syntax:

It ensures that each row for a column must have a unique value. It is like a Primary key but it can accept only one null value. In a table one or more column can contain a Unique Constraint.

Syntax:

Column Level:

```
CREATE TABLE tableName (  
  
    columnName datatype CONSTRAINT constraintName UNIQUE  
  
);
```

Table Level:

```
ALTER TABLE tableName  
  
ADD CONSTRAINT constraintName UNIQUE(columnName)
```

PRIMARY Constraint Syntax:

A Primary key uniquely identifies each row in a table. It cannot accept null and duplicate data. One or more of the columns of a table can contain a Primary key.

Syntax:

Column Level:

```
CREATE TABLE tableName (  
  
columnName datatype CONSTRAINT constraintName PRIMARY KEY  
  
);
```

Table Level:

```
ALTER TABLE tableName  
  
ADD CONSTRAINT constraintName PRIMARY KEY(columnName)
```

FOREIGN Constraint Syntax:

A Foreign Key is a field in a database table that is a Primary key in another table. A Foreign key creates a relation between two tables. The first table contains a primary key and the second table contains a foreign key.

Syntax:

Column Level:

```
CREATE TABLE tableName (  
  
columnName datatype CONSTRAINT constraintName REFERENCES  
referenceTableName (referenceColumnName)  
  
);
```

Table Level:

ALTER TABLE tableName

ADD CONSTRAINT constraintName FOREIGN KEY(columnName)

REFERENCES referenceTableName(referenceColumnName);

Good luck! Find the lab task on the next page of the manual.

Class Task

Consider the database schema of Facebook mentioned below

User

id	name	username	DOB
----	------	----------	-----

- The attribute 'id' shall be of type integer, not null, must have specific limit and is a primary key.
- The attribute 'name' shall be of varchar type, shall not be null, have specific limit.
- The attribute 'username' shall be alphanumeric, shall not be null.
- DOB stands for Date of Birth, it shall be of UNIXTIME and can be null.

Post

id	description	date	user_id
----	-------------	------	---------

- The attribute 'id' shall be of type integer, not null, must have specific limit and is a primary key.
- The attribute 'description' shall be of varchar type, shall not be null.
- The attribute 'date' shall store the time of a post in the form UNIXTIME.
- The attribute 'user_id' shall be a value from table user and shall have FOREIGN KEY constraint.

Comment

id	description	date	post_id	user_id
----	-------------	------	---------	---------

- The attribute 'id' shall be of type integer, not null, must have specific limit and is a primary key.
- The attribute 'description' shall be varchar type, shall not be null.

- The attribute 'date' shall store the time of a comment in the form UNIXTIME.
- The attribute 'post_id' shall be a value from Post table and the 'user_id' shall be from the user table. 'post_id' is supposed to describe the post on which the comment is done while the 'user_id' is supposed to describe the user who commented on the post. Note, both of the attributes i.e., post_id and user_id shall have FOREIGN_KEY constraint.