SQL – ALTER TABLE Statement

The SQL **ALTER TABLE** command is a part of Data Definition Language (DDL) and modifies the structure of a table. The ALTER TABLE command can add or delete columns, create or destroy indexes, change the type of existing columns, or rename columns or the table itself.

The ALTER TABLE command can also change characteristics of a table such as the storage engine used for the table. We will make use of the following table in our examples

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00

Syntax

Following is the basic syntax of an ALTER TABLE command –

```
ALTER TABLE table_name [alter_option ...];
```

Where, the **alter_option** depends on the type of operation to be performed on a table. This article will discuss such important operations one by one.

ALTER TABLE – ADD Column

If you need to add a new column to a table, you should use the **ADD COLUMN** option along with ALTER TABLE statement as shown below –

```
ALTER TABLE table_name ADD column_name datatype;
```

Following is the example to ADD a New Column to an existing table –

```
ALTER TABLE CUSTOMERS ADD SEX char(1);
```

Verification

To verify whether the CUSTOMERS table is altered by adding a new column SEX, use the SELECT statement to retrieve the records of the table –

```
SELECT * FROM CUSTOMERS;
```

ALTER TABLE – DROP COLUMN

If you need to drop an existing column from a table, you should use the DROP COLUMN option along with ALTER TABLE statement as shown below.

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Example

Following is the example to DROP sex column from the existing table.

```
ALTER TABLE CUSTOMERS DROP COLUMN SEX;
```

```
SELECT * FROM CUSTOMERS;
```

ALTER TABLE – ADD INDEX

You can add index to an existing column of a table using the ADD INDEX statement along with the ALTER statement –

```
ALTER TABLE table_name
ADD INDEX index_name [index_type]
```

Example

Following query adds an index on the column NAME of CUSTOMERS table –

```
ALTER TABLE CUSTOMERS ADD INDEX name_index (NAME);
```

ALTER TABLE – DROP INDEX

You can drop an existing index from a table using the DROP INDEX statement along with the ALTER statement –

```
ALTER TABLE table_name DROP INDEX index_name;
```

Example

Following query adds an index on the column NAME of CUSTOMERS table –

```
ALTER TABLE CUSTOMERS DROP INDEX name_index;
```

ALTER TABLE – ADD PRIMARY KEY

Following is the syntax to add a primary key in an existing table of a database −

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name
PRIMARY KEY (column1, column2...);
```

Example

Before we add a primary key to an existing table, first let's create a new table called EMPLOYEES as follows**:**

```
CREATE TABLE EMPLOYEES(
   ID         INT NOT NULL,
   NAME       VARCHAR (20) NOT NULL,
   AGE        INT NOT NULL,
   ADDRESS    CHAR (25),
   SALARY     DECIMAL (18, 2)
);
```

Following query adds primary key constraint on the column ID of EMPLOYEES table –

```
ALTER TABLE EMPLOYEES
ADD CONSTRAINT MyPrimaryKey
PRIMARY KEY(ID);
```

Verification

To verify the above query if you describe the table using the DESC EMPLOYEES command −

```
DESC EMPLOYEES;
```

This will display the structure of the table created: column names, their respective data types, constraints (if any) etc.

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI	NULL	
NAME	varchar(20)	NO		NULL	
AGE	int(11)	NO		NULL	
ADDRESS	char(25)	YES		NULL	
SALARY	decimal(18,2)	YES		NULL	

ALTER TABLE – DROP PRIMARY KEY

Following is the syntax to delete a primary key from an existing table of a database –

```
ALTER TABLE table_name DROP PRIMARY KEY;
```

Example

Following query deletes primary key constraint from the column ID of EMPLOYEES table –

```
ALTER TABLE EMPLOYEES DROP PRIMARY KEY;
```

ALTER TABLE – ADD CONSTRAINT

Following is the syntax to add a unique constraint to a column of an existing table −

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name
UNIQUE(column1, column2...);
```

Example

Following query adds UNIQUE constraint to the table CUSTOMERS –

```
ALTER TABLE EMPLOYEES ADD CONSTRAINT CONST UNIQUE(NAME);
```

ALTER TABLE – DROP CONSTRAINT

Following is the syntax to drop a unique constraint from an existing table –

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

Example

Following query adds UNIQUE constraint to the table CUSTOMERS –

```
ALTER TABLE EMPLOYEES DROP CONSTRAINT CONST;
```

ALTER TABLE – RENAME COLUMN

Following is the syntax to rename a column name of an existing table –

```
ALTER TABLE table_name
RENAME COLUMN old_column_name to new_column_name;
```

Example

Following query renames NAME column in table CUSTOMERS −

```
ALTER TABLE CUSTOMERS RENAME COLUMN name to full_name;
```

ALTER TABLE – MODIFY DATATYPE

Following is the syntax to change the data type of any column in MySQL, MS Server and Oracle.

SQL Server/MS Access Syntax

```
ALTER TABLE table_name ALTER COLUMN column_name datatype;
```

MySQL Syntax

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

Oracle Syntax

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

Example

Following query modifies datatype of SALARY column in MySQL CUSTOMERS table −

```
ALTER TABLE CUSTOMERS MODIFY COLUMN ID DECIMAL(18, 4);
```