**DDL COMMAND**

DDL or Data Definition Language consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

* [**CREATE**](https://www.geeksforgeeks.org/sql-create/) – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
* [**DROP**](https://www.geeksforgeeks.org/sql-drop-truncate/) – is used to delete objects from the database.
* [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)-is used to alter the structure of the database.
* [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)–is used to remove all records from a table, including all spaces allocated for the records are removed.
* [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/) –is used to add comments to the data dictionary.
* [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/) –is used to rename an object existing in the database.

**CREATE**

There are two CREATE statements available in SQL:

1. CREATE DATABASE
2. CREATE TABLE

A Database is defined as a structured set of data. So, in SQL the very first step to store the data in a well-structured manner is to create a database. The CREATE DATABASE statement is used to create a new database in SQL.

Syntax:

CREATE DATABASE database\_name;

database\_name: name of the database.

Example Query:  
This query will create a new database in SQL and name the database as my\_database.

CREATE DATABASE my\_database;

The CREATE TABLE statement is used to create a table in SQL. Table comprises of rows and columns. So, while creating tables we must provide all the information to SQL about the names of the columns, type of data to be stored in columns, size of the data etc. Let us now dive into details on how to use CREATE TABLE statement to create tables in SQL.

Syntax:

CREATE TABLE table\_name

(

column1 data\_type(size),

column2 data\_type(size),

column3 data\_type(size),

....

);

**table\_name**: name of the table.

**column1** name of the first column.

**data\_type**: Type of data we want to store in the column.

For example,int for integer data.

**size**: Size of the data we can store in a column. For example, if for

a column we specify the datatype as int and size as 10 then this column can store an integer

number of maximum 10 digits.

**Example Query:**  
This query will create a table named Students with three columns, ROLL\_NO, NAME and SUBJECT.

CREATE TABLE Students

(

ROLL\_NO int(3),

NAME varchar(20),

SUBJECT varchar(20),

);

**DROP**

The Oracle **DROP TABLE statement** allows you to remove or delete a table from the Oracle database.

**Syntax**

The syntax for the Oracle DROP TABLE statement is:

DROP TABLE [schema\_name].table\_name

[ CASCADE CONSTRAINTS ]

[ PURGE ];

Parameters or Arguments

**schema\_name**

The name of the schema that owns the table.

**table\_name**

The name of the table to remove from the Oracle database.

CASCADE CONSTRAINTS

Optional. If specified, all referential integrity constraints will be dropped as well.

**PURGE**

Optional. If specified, the table and its dependent objects will be purged from the recycle bin and you will not be able to recover the table. If not specified, the table and its dependent objects are placed in the recycle bin and can be recovered later, if needed.

**Note**

* If there are referential integrity constraints on table\_name and you do not specify the CASCADE CONSTRAINTS option, the DROP TABLE statement will return an error and Oracle will not drop the table.

example

DROP TABLE customers;

This Oracle DROP TABLE example would drop the table called *customers*.

### Purge

Let's look at how to use the PURGE option with the DROP TABLE statement in Oracle.

When issuing a DROP TABLE statement in Oracle, you can specify the PURGE option. The PURGE option will purge the table and its dependent objects so that they do not appear in the recycle bin. The risk of specifying the PURGE option is that you will not be able to recover the table. However, the benefit of using PURGE is that you can ensure that sensitive data will not be left sitting in the recycle bin.

**For example:**

DROP TABLE customers PURGE;

This DROP TABLE statement would drop the table called *customers* and issue a PURGE so that the space associated with the *customers* table is released. In other words, the *customers* table is not placed into the recycle bin and, therefore, cannot be recovered later if required.

WE can drop other database objects

DROP object object\_name;

Drop procedure test\_proc;

**TRUNCATE**

The TRUNCATE TABLE statement is used to remove all records from a table in Oracle. It performs the same function as a DELETE statement without a WHERE clause.

**Warning**: If you truncate a table, the TRUNCATE TABLE statement cannot be rolled back.

## Syntax

The syntax for the TRUNCATE TABLE statement in Oracle/PLSQL is:

TRUNCATE TABLE [schema\_name.]table\_name

[ PRESERVE MATERIALIZED VIEW LOG | PURGE MATERIALIZED VIEW LOG ]

[ DROP STORAGE | REUSE STORAGE ] ;

### Parameters or Arguments

**schema\_name**

Optional. If specified, it is the name of the schema that the table belongs to.

**table\_name**

The table that you wish to truncate.

**PRESERVE MATERIALIZED VIEW LOG**

**Optional**. If specified, the materialized view log will be preserved when the table is truncated. This is the default behavior.

**PURGE MATERIALIZED VIEW LOG**

Optional. If specified, the materialized view log will be purged when the table is truncated.

**DROP STORAGE**

Optional. If specified, all storage for the truncated rows will be deallocated, except the space that has been allocated by MINEXTENTS. This is the default behavior.

**REUSE STORAGE**

Optional. If specified, all storage for the truncated rows will remain allocated to the table.

## Example

In Oracle, truncating a table is a fast way to clear out records from a table if you don't need to worry about rolling back. One of the reasons is that when the table is truncated, it does not affect any of the table's indexes, triggers, or dependencies. Truncating a table is also a lot easier than dropping the table and recreating it.

Let's look at an example of how to use the TRUNCATE TABLE statement in Oracle/PLSQL.

**For example:**

TRUNCATE TABLE customers;

This example would truncate the table called *customers* and remove all records from that table.

It would be equivalent to the following DELETE statement in Oracle:

DELETE FROM customers;

Both statements would result in all data from the *customers* table being deleted. The main difference between the two is that you can roll back the DELETE statement if you choose, but you can't roll back the TRUNCATE TABLE statement.

**ALTER/RENAME**

The Oracle ALTER TABLE statement is used to add, modify, or drop/delete columns in a table. The Oracle ALTER TABLE statement is also used to rename a table.

Example that shows how to add a column in an Oracle table using the ALTER TABLE statement.

ALTER TABLE customers

ADD customer\_name varchar2(45);

we could use the ALTER TABLE statement to add a new column that also has a default value:

ALTER TABLE customers

ADD city varchar2(40) DEFAULT 'Seattle';

To ADD MULTIPLE COLUMNS to an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table\_name

ADD (column\_1 column\_definition,

column\_2 column\_definition,

...

column\_n column\_definition);

ALTER TABLE customers

ADD (customer\_name varchar2(45),

city varchar2(40) DEFAULT 'Seattle');

To MODIFY A COLUMN in an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table\_name

MODIFY column\_name column\_type;

how to modify a column in an Oracle table using the ALTER TABLE statement.

For example:

ALTER TABLE customers

MODIFY customer\_name varchar2(100) NOT NULL;

ALTER TABLE customers

MODIFY city varchar2(75) DEFAULT 'Seattle' NOT NULL;

Example shows how to modify multiple columns in an Oracle table using the ALTER TABLE statement.

For example:

ALTER TABLE customers

MODIFY (customer\_name varchar2(100) NOT NULL,

city varchar2(75) DEFAULT 'Seattle' NOT NULL);

To DROP A COLUMN in an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table\_name

DROP COLUMN column\_name;

ALTER TABLE customers

DROP COLUMN customer\_name;

To RENAME A COLUMN in an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table\_name

RENAME COLUMN old\_name TO new\_name;

ALTER TABLE customers

RENAME COLUMN customer\_name TO cname;

## Rename table

### Syntax

To RENAME A TABLE, the Oracle ALTER TABLE syntax is:

ALTER TABLE table\_name

RENAME TO new\_table\_name;

### Example

Example shows how to rename a table in Oracle using the ALTER TABLE statement.

ALTER TABLE customers

RENAME TO contacts;

**COMMENT**

Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.

Single line comments start with –

--Select all:

SELECT \* FROM Customers;

Multi-line comments start with /\* and end with \*/.

/\*Select all the columns  
of all the records  
in the Customers table:\*/  
SELECT \* FROM Customers;

We can use to comment to write comment on table or column to know its description.

COMMENT ON TABLE Employee IS 'This is a table for Employee.';

COMMENT ON COLUMN product.product\_description

IS 'This is comment for the column';

**Data dictionary for tables ,comments are**

**For comments**

select \* from user\_col\_comments where table\_name = 'COM';

TABLE\_NAME                     COLUMN\_NAME                    COMMENTS

------------------------------ ------------------------------ --------------------

COM                            A                              a test

**For tables**

Select \* from user\_Tables;

Select \* from all\_tables;