

**SQL Structured Query Language**

SQL is a standard language for storing, manipulating and retrieving data in databases.

What is SQL?

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* **SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987**

## What Can SQL do?

* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database
* SQL can create stored procedures in a database
* SQL can create views in a database
* SQL can set permissions on tables, procedures, and views

SQL is not a database system, but it is a query language.

Suppose you want to perform the queries of SQL language on the stored data in the database. You are required to install any database management system in your systems, for example, [**Oracle**](https://www.javatpoint.com/oracle-tutorial)**,**[**MySQL**](https://www.javatpoint.com/mysql-tutorial)**,**[**PostgreSQL**](https://www.javatpoint.com/postgresql-tutorial)**,**[**SQL Server**](https://www.javatpoint.com/sql-server-tutorial)**,**[**DB2**](https://www.javatpoint.com/db2-tutorial), etc

**Oracle DB**

**Oracle Corporation** is the largest software company to develop and markets computer software applications for business. The company is best known for its Oracle database products and, more recently, cloud products and services

Oracle database is a relational database management system. It is also called **OracleDB**, or simply **Oracle**. It is produced and marketed by **Oracle Corporation**. It was created in **1977** by **Lawrence Ellison** and other engineers. It is one of the most popular relational database engines

* Oracle is a relational database management system. It is widely used in enterprise applications.
* Oracle database such as insert record, update record, delete record, select record, create table, drop table etc.

Oracle Data Types

**Oracle String data types**

|  |  |
| --- | --- |
| **CHAR(size)** | It is used to store character data within the predefined length. It can be stored up to 2000 bytes. |
| **NCHAR(size)** | It is used to store national character data within the predefined length. It can be stored up to 2000 bytes. |
| **VARCHAR2(size)** | It is used to store variable string data within the predefined length. It can be stored up to 4000 byte. |
| **VARCHAR(SIZE)** | It is the same as VARCHAR2(size). You can also use VARCHAR(size), but it is suggested to use VARCHAR2(size) |
| **NVARCHAR2(size)** | It is used to store Unicode string data within the predefined length. We have to must specify the size of NVARCHAR2 data type. It can be stored up to 4000 bytes. |

**Oracle Numeric Data Types**

|  |  |
| --- | --- |
| **NUMBER(p, s)** | It contains precision p and scale s. The precision p can range from 1 to 38, and the scale s can range from -84 to 127. |
| **FLOAT(p)** | It is a subtype of the NUMBER data type. The precision p can range from 1 to 126. |
| **BINARY\_FLOAT** | It is used for binary precision( 32-bit). It requires 5 bytes, including length byte. |
| **BINARY\_DOUBLE** | It is used for double binary precision (64-bit). It requires 9 bytes, including length byte. |

**Oracle Date and Time Data Types**

|  |  |
| --- | --- |
| **DATE** | It is used to store a valid date-time format with a fixed length. Its range varies from January 1, 4712 BC to December 31, 9999 AD. |
| **TIMESTAMP** | It is used to store the valid date in YYYY-MM-DD with time hh:mm:ss format. |

**Oracle Large Object Data Types (LOB Types)**

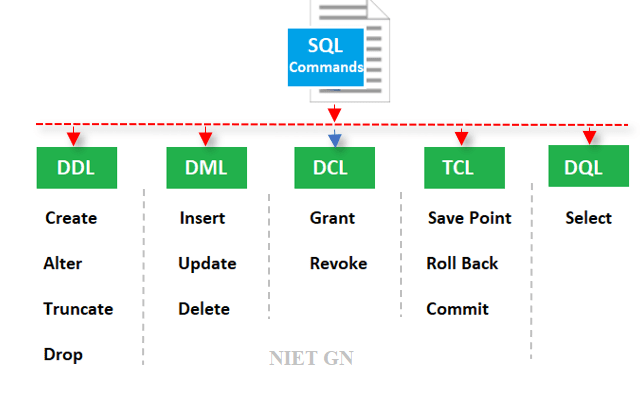
|  |  |
| --- | --- |
| **BLOB** | It is used to specify unstructured binary data. Its range goes up to 232-1 bytes or 4 GB. |
| **BFILE** | It is used to store binary data in an external file. Its range goes up to 232-1 bytes or 4 GB. |
| **CLOB** | It is used for single-byte character data. Its range goes up to 232-1 bytes or 4 GB. |
| **NCLOB** | It is used to specify single byte or fixed length multibyte national character set (NCHAR) data. Its range is up to 232-1 bytes or 4 GB. |
| **RAW(size)** | It is used to specify variable length raw binary data. Its range is up to 2000 bytes per row. Its maximum size must be specified. |
| **LONG RAW** | It is used to specify variable length raw binary data. Its range up to 231-1 bytes or 2 GB, per row. |

# **SQL Commands in Oracle database**

SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

## Types of SQL Commands

There are five types of SQL commands: **DDL, DML, DCL, TCL, and DQL.**



* DDL (Data Definition Language)
* DML (Data Manipulation Language)
* DCL (Data Control Language)
* TCL (Transaction Control Language)
* DQL (Data Query Language) (DRL)

**DDL (Data Definition Language)**

1. CREATE Command
2. DROP Command
3. ALTER Command
4. TRUNCATE Command
5. RENAME Command

**CREATE:** This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).

**DROP:** This command is used to delete objects from the database.

**ALTER:** This is used to alter the structure of the database.

**TRUNCATE:** This is used to remove all records from a table, including all spaces allocated for the records are removed.

**COMMENT:** This is used to add comments to the data dictionary.

**RENAME:** This is used to rename an object existing in the database.

**CREATE Command**

CREATE is a DDL command its used to create databases, tables, triggers and other database objects.

1. The Table name must begin with a letter A-Z or a-z
2. The Table name can contain numbers and underscores
3. The name of the table can be in UPPER or lower case
4. The maximum length of the table name can be 30 characters
5. We cannot use the same name of another existing object in our schema
6. Don’t provide space in the table name. If you want to provide space in a table name then you can use the underscore symbol.
7. A table should contain a minimum of 1 column and a maximum of 1000 columns.

**CREATE DDL Command in Oracle**:

**CREATE** **TABLE** table\_name

(

column\_Name1 data\_type ( **size** **of** the **column** ) ,

column\_Name2 data\_type ( **size** **of** the **column**) ,

column\_Name3 data\_type ( **size** **of** the **column**) ,

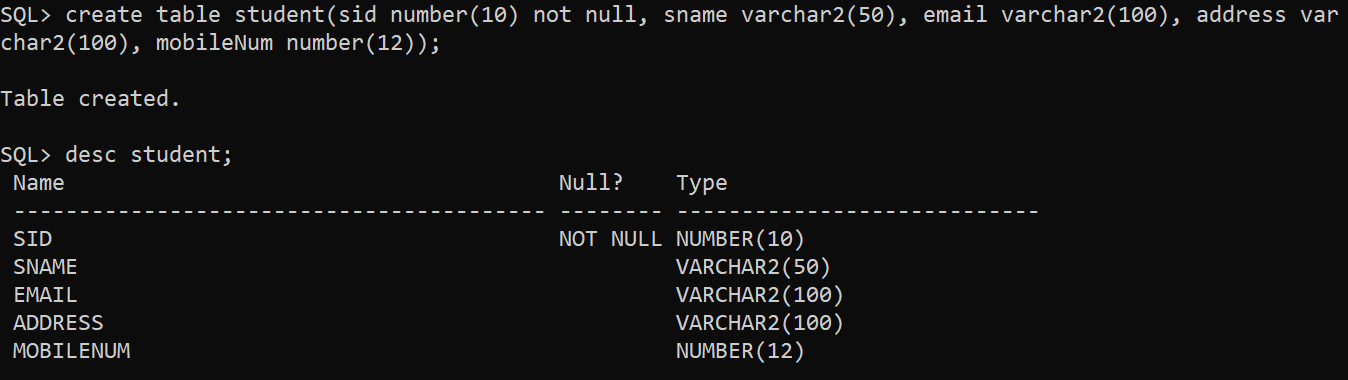
...

column\_NameN data\_type ( **size** **of** the **column** )

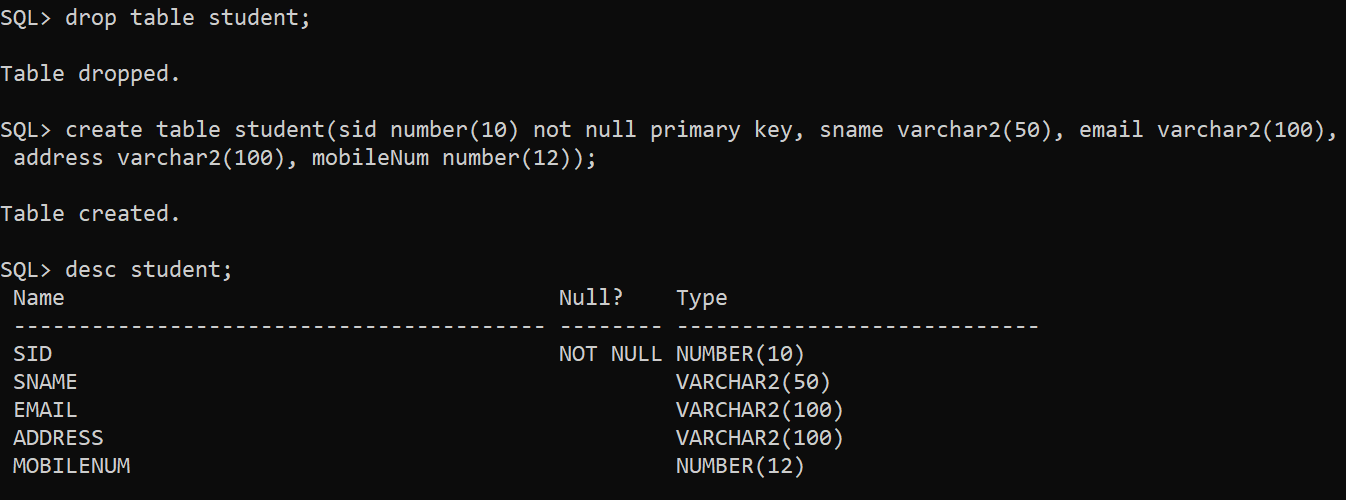
) ;

**create table student(sid number(10), sname varchar2(50), email varchar2(100), address varchar2(100), mobileNum number(12));**

**create table student(sid number(10) not null, sname varchar2(50), email varchar2(100), address varchar2(100), mobileNum number(12));**



**create table student(Sid number(10) not null primary key, same varchar2(50), email varchar2(100), address varchar2(100), mobileNum number(12));**



#### **create table student(sid number(10), sname varchar2(50), email varchar2(100), address varchar2(100), mobileNum number(12), admsiodate Date);**

#### 

1. **CREATE** **TABLE** Student
2. (
3. Roll\_No. number(10) NOT NULL,
4. First\_Name varchar2 (20) ,
5. Last\_Name varchar2 (20) ,
6. Age **Int** ,
7. Marks **Int** ,
8. ) ;

CREATETABLE persons ( idINTNOTNULLPRIMARYKEYAUTO\_INCREMENT, name VARCHAR(50)NOTNULL,birth\_date DATE, phone VARCHAR(15)NOTNULLUNIQUE);

CREATETABLE persons ( idINTNOTNULLPRIMARYKEYIDENTITY(1,1), name VARCHAR(50)NOTNULL,birth\_dateDATE, phone VARCHAR(15)NOTNULLUNIQUE);

## PRIMARY KEY on CREATE TABLE

he PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key;

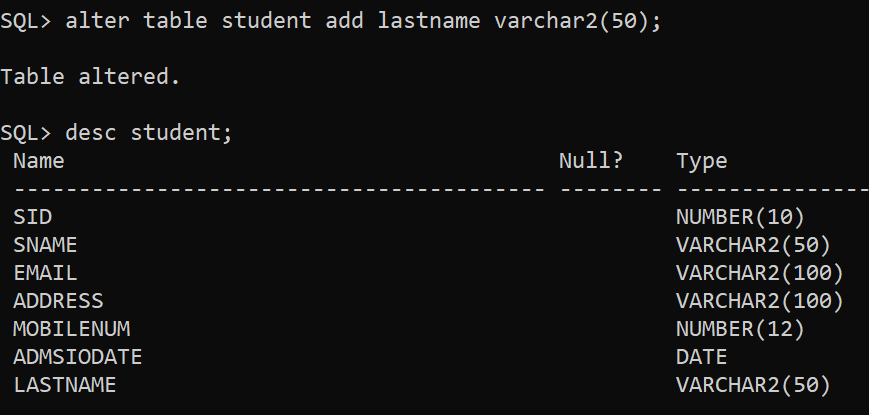
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (ID)  
);

CREATE TABLE Persons (  
    ID int NOT NULL PRIMARY KEY,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);

* The NOT NULL constraint ensures that the field cannot accept a NULL value.
* The PRIMARY KEY constraint marks the corresponding field as the table's primary key.
* The AUTO\_INCREMENT attribute is a SQL extension to standard SQL, which tells MySQL to automatically assign a value to this field if it is left unspecified, by incrementing the previous value by 1. Only available for numeric fields.
* The UNIQUE constraint ensures that each row for a column must have a unique value.

ALTER  **DDL Command in Oracle:**

1. Increase/decrease the length of a column.
2. Change the data type of a column.
3. Change NOT NULL to NULL or NULL to NOT NULL.
4. Used to add a new column to an existing table.
5. Used to drop an existing column.
6. alter table student add lastname varchar2(50);



1. **ALTER** **TABLE** name\_of\_table **ADD** column\_name column\_definition;

**ALTER** **TABLE** Student **ADD** Father's\_Name **Varchar**(60);

1. **ALTER** **TABLE** customers   **ADD** customer\_age varchar2(50);

**RENAME**

RENAME : rename an objects

**RENAME <OLD TABLE NAME> TO <NEW TABLE NAME>;**

**RENAME EMPLOYEE TO EMPLOYEEDETAILS;**

**TRUNCATE**

**TRUNCATE** delete all the records or rows from a table without any condition,

**TRUNCATE** **TABLE** customers;

**TRUNCATE TABLE <TABLE NAME>;**

**TRUNCATE TABLE EMPLOYEE;**

##### **DROP**

**DROP** **TABLE** Table\_Name;

**DROP** **TABLE** Student;

**DROP TABLE EMPLOYEE;**

**DROP** **TABLE** customers;

**Note**

##### **How to view the list of tables in the oracle database?**

To view the list of tables in the Oracle database, we need to use the following syntax.

**SELECT \* FROM TAB;**

##### **How to view the structure of a table in the oracle database?**

**Syntax: DESC <TABLE NAME>;** (Desc = Describe)

**Example: DESC EMPLOYEE;**

**DESCstudents;**

# **DML Commands in SQL**

1. INSERT Command
2. UPDATE Command
3. DELETE Command

### **DML(Data Manipulation Language):**

1. **INSERT**– Is used to insert data into a table.
2. **UPDATE**– Is used to update existing data within a table.
3. **DELETE**– Is used to delete records from a database table.
4. **MERGE**– Is used to merge records of tables.

# **INSERT Statement in Oracle**

In Oracle, INSERT statement is used to add a single record or multiple records into the table.

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

**INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');**

**INSERT INTO Customers (CustomerName, City, Country) VALUES ('Cardinal', 'Stavanger', 'Norway');**

INSERT INTO employee (employee\_id,name,vehicle\_name) VALUES('AD010','Sharmishtha', 'Hector');

INSERT INTO suppliers (supplier\_id, supplier\_name) VALUES (50, 'Flipkart');

**INSERT INTO students (student\_id, student\_name, student\_age) VALUES (3, 'Happy’, 11');**

INSERT INTO persons (name,birth\_date, phone)VALUES('Peter Wilson','1990-07-15','0711-020361');

### **DQL (Data Query Language):DRL**

* [**SELECT**](https://www.geeksforgeeks.org/sql-select-clause/)**:**It is used to retrieve data from the database.

SELECT\*FROM employees;

SELECTemp\_id,emp\_name,hire\_date, salary FROM employees;

SELECT CustomerName, City FROM Customers;

## The IS NULL Operator

The IS NULL operator is used to test for empty values (NULL values).

The following SQL lists all customers with a NULL value in the "Address" field:

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NULL;

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NULL;

## The IS NOT NULL Operator

The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).

The following SQL lists all customers with a value in the "Address" field:

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NOT NULL;

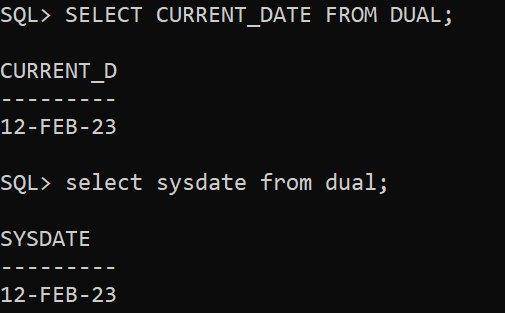
SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NOT NULL;

#### CURRENT\_DATE

SELECT CURRENT\_DATE FROM DUAL;

#### SYSDATE

select sysdate from dual;



**DISTINCT**

The DISTINCT clause is used to remove duplicate rows from the result set:

SELECT DISTINCT *column\_list* FROM *table\_name*;

SELECT city FROM customers;

SELECTDISTINCT city FROM customers;

SELECT DISTINCT statement for a column that has multiple NULL values, Then SQL keeps one NULL value and removes others from the result set, because DISTINCT treats all the NULL values as the same value.

**UPDATE Oracle**

The UPDATE statement is used to modify the existing records in a table

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;

UPDATE suppliers SET supplier\_name = 'Kingfisher' WHERE supplier\_id = 2;

UPDATE suppliers SET supplier\_address = 'Agra', supplier\_name = 'Bata shoes' WHERE supplier\_id = 1;

**Oracle DELETE Statement**

In Oracle, DELETE statement is used to remove or delete a single record or multiple records from a table.

DELETEFROM*table\_name*WHERE*condition*;

DELETEFROM persons WHERE id >3;

DELETEFROM persons;

DELETE FROM Customers;

DELETE FROM Customers WHERE CustomerName='AlfredsFutterkiste';

### **DCL (Data Control Language):**

* [**GRANT:**](https://www.geeksforgeeks.org/mysql-grant-revoke-privileges/)This commandgives users access privileges to the database.
* [**REVOKE:**](https://www.geeksforgeeks.org/difference-between-grant-and-revoke/)This command withdraws the user’s access privileges given by using the GRANT command.

### **TCL (Transaction Control Language):**

* [**COMMIT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Commits a Transaction.
* [**ROLLBACK**](https://www.geeksforgeeks.org/sql-transactions/)**:**Rollbacks a transaction in case of any error occurs.
* [**SAVEPOINT**](https://www.geeksforgeeks.org/sql-transactions/)**:**Sets a save point within a transaction.

# **Oracle Insert Statement**

In Oracle, INSERT statement is used to add a single record or multiple records into the table.

**Syntax: (Inserting a single record using the Values keyword):**

INSERT INTO table\_name (column\_list) VALUES( value\_list);

1. **INSERT** **INTO** **table**
2. (column1, column2, ... column\_n )
3. **VALUES**
4. (expression1, expression2, ... expression\_n );

**Syntax: (Inserting multiple records using a SELECT statement):**

1. **INSERT** **INTO** suppliers  (supplier\_id, supplier\_name)  **VALUES**  (50, 'Flipkart');

CREATE TABLE employee\_details(

emp\_idVARCHAR(8),

emp\_nameVARCHAR(20),

emp\_dept\_idVARCHAR(20),

emp\_age INT);

CREATE TABLE discounts (

discount\_id NUMBER GENERATED BY DEFAULT AS IDENTITY,

discount\_name VARCHAR2(255) NOT NULL,

amount NUMBER(3,1) NOT NULL,

start\_date DATE NOT NULL,

expired\_date DATE NOT NULL

);

INSERT INTO discounts(discount\_name, amount, start\_date, expired\_date)

VALUES('Summer Promotion', 9.5, DATE '2017-05-01', DATE '2017-08-31');

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

INSERT INTO Customers (CustomerName, City, Country)  
VALUES ('Cardinal', 'Stavanger', 'Norway');

SELECT\*FROMdiscounts;

Exa1

create table Info(id integer, Cost integer, city varchar(200));

insert into Info(id, Cost,city) values(1, 100,"Pune");

insert into Info(id, Cost,city) values(2, 50, "Satara");

insert into Info(id, Cost,city) values(3, 65,"Pune");

insert into Info(id, Cost,city) values(4, 97,"Mumbai");

insert into Info(id, Cost,city) values(5, 12,"USA");

select \* from Info;

Exa2

create table Info(id integer, Cost integer,citynvarchar(200));

INSERT INTO Info (id,Cost,city)

VALUES (1,200, 'Pune'), (2, 150,'USA'), (3,345, 'France');

INSERT INTO Orders\_TblVALUES('1250', '2011-10-21', '8740', 'Not Specified', '35');

INSERT INTO Orders\_TblVALUES('1251', '2011-11-21', '8741', 'Not Specified', '36');

INSERT INTO Orders\_TblVALUES('1252', '2011-12-21', '8742', 'Not Specified', '37');

INSERT INTO Pets (PetId, PetTypeId, OwnerId, PetName, DOB)

VALUES ( 1, 2, 3, 'Fluffy', '2020-11-20' );

INSERT INTO Pets (PetId, PetTypeId, OwnerId, PetName, DOB)

VALUES ( 2, 3, 3, 'Fetch', '2019-08-16' );

INSERT INTO Pets (PetId, PetTypeId, OwnerId, PetName, DOB)

VALUES ( 3, 2, 2, 'Scratch', '2018-10-01' );

INSERT INTO Pets (PetId, PetTypeId, OwnerId, PetName, DOB)

VALUES

(1, 2, 3, 'Fluffy', '2020-11-20'),

(2, 3, 3, 'Fetch', '2019-08-16'),

(3, 2, 2, 'Scratch', '2018-10-01');

select \* from Info;

**UPDATE**

he UPDATE statement is used to modify the existing records in a table.

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

## Oracle Update Example: (Update single column)

1. **UPDATE** suppliers
2. **SET** supplier\_name = 'Kingfisher'
3. **WHERE** supplier\_id = 2;

## Oracle Update Example: (Update multiple columns)

1. **UPDATE** suppliers
2. **SET** supplier\_address = 'Agra',
3. supplier\_name = 'Bata shoes'
4. **WHERE** supplier\_id = 1;

## DELETE Statement

The DELETE statement is used to delete existing records in a table.

### **DELETE Syntax**

DELETE FROM table\_name WHERE condition;

DELETE FROM Customers WHERE CustomerName='AlfredsFutterkiste';

1. **DELETE** **FROM** customers  **WHERE** **name** = 'Sohan';
2. **DELETE** **FROM** customers  **WHERE** last\_name = 'Maurya'  AND customer\_id > 2;

Delete All Records

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM *table\_name*;

DELETE FROM Customers;

| **Delete** | **Truncate** |
| --- | --- |
| The DELETE command is used to delete specified rows(one or more). | While this command is used to delete all the rows from a table. |
| It is a DML(Data Manipulation Language) command. | While it is a DDL(Data Definition Language) command. |
| There may be a WHERE clause in the DELETE command in order to filter the records. | While there may not be WHERE clause in the TRUNCATE command. |
| In the DELETE command, a tuple is locked before removing it. | While in this command, the data page is locked before removing the table data. |
| The DELETE statement removes rows one at a time and records an entry in the transaction log for each deleted row. | TRUNCATE TABLE removes the data by deallocating the data pages used to store the table data and records only the page deallocations in the transaction log. |
| DELETE command is slower than TRUNCATE command. | While the TRUNCATE command is faster than the DELETE command. |
| To use Delete you need DELETE permission on the table. | To use Truncate on a table we need at least ALTER permission on the table. |
| The identity of the fewer column retains the identity after using DELETE Statement on the table. | Identity the column is reset to its seed value if the table contains an identity column. |
| The delete can be used with indexed views. | Truncate cannot be used with indexed views. |
| This command can also active trigger. | This command does not active trigger. |
| DELETE statement occupies more transaction spaces than Truncate. | Truncate statement occupies less transaction spaces than DELETE. |

## Example of DELETE, DROP, and TRUNCATE commands in SQL

* The SQL DROP command is a DDL ( Data Definition Language ) command that deletes the defined table with all its table data, associated indexes, constraints, triggers, and permission specifications.
* The SQL DELETE command is a DML ( Data Manipulation Language ) command that deletes existing records from the table in the database.
* The SQL TRUNCATE command is a DDL ( Data Definition Language ) command that modifies the data in the database. The TRUNCATE command helps us delete the complete records from an existing table in the database.

DROP TABLE Student;

DELETE FROM Student WHERE Roll\_No = 101;

TRUNCATE TABLE Student;

DQL

# **Oracle SELECT Statement**

The Oracle SELECT statement is used to retrieve data from one or more than one tables, object tables, views, object views etc.

SELECT column1, column2, ...  
FROM table\_name;

SELECT \* FROM table\_name;

SELECT \* FROM Customers;

SELECT CustomerName, City FROM Customers;

# **SELECT UNIQUE**

Actually, there is no difference between DISTINCT and UNIQUE.

**SELECT UNIQUE** is an old syntax which was used in oracle description but later ANSI standard defines DISTINCT as the official keyword.

After that oracle also added DISTINCT but did not withdraw the service of UNIQUE keyword for the sake of backward compatibility.

In simple words, we can say that SELECT UNIQUE statement is used to retrieve a unique or distinct element from the table.

1. **SELECT** **UNIQUE** column\_name
2. **FROM** table\_name;

# **SELECT DISTINCT**

1. **DISTINCT command** is used with SELECT key word to retrieve only distinct or unique data.
2. In a table, there may be a chance to exist a duplicate value and sometimes we want to retrieve only unique values. In such scenarios, SQL SELECT DISTINCT statement is used
3. **SELECT** **DISTINCT** expressions  **FROM** tables  **WHERE** conditions;
4. **CREATE** **TABLE**  "CUSTOMERS"
5. (    "NAME" VARCHAR2(4000),
6. "AGE" NUMBER,
7. "SALARY" NUMBER,
8. "STATE" VARCHAR2(4000)
9. )
10. **SELECT** **DISTINCT** state
11. **FROM** customers
12. **WHERE** **name** = 'charu';
13. **SELECT** **DISTINCT** **name**, age, salary
14. **FROM** customers
15. **WHERE** age >= '60';

#### **SQL SELECT UNIQUE and SQL SELECT DISTINCT statements are same.**

# **ORDER BY Clause**

In Oracle, ORDER BY Clause is used to sort or re-arrange the records in the result set. The ORDER BY clause is only used with SELECT statement.

1. **SELECT** \*
2. **FROM** customers
3. **WHERE** salary >= 20000
4. **ORDER** **BY** salary **ASC**;

**ASC:** It is an optional parameter that is used to sort records in ascending order.

**DESC:** It is also an optional parameter that is used to sort records in descending order

1. **SELECT** \*
2. **FROM** supplier
3. **ORDER** **BY** last\_name;

# **SQL SELECT COUNT**

The **SQL COUNT()** is a function that returns the number of records of the table in the output.

This function is used with the SQL SELECT statement.

## Select Count(\*) Function in SQL

The count(\*) function in SQL shows all the Null and Non-Null records present in the table.

### **Syntax of Count (\*) Function in SQL**

1. **SELECT** COUNT(\*) **FROM** table\_name;
2. **SELECT** COUNT (\*)  **FROM** Bikes ;
3. **SELECT** COUNT(column\_name) **FROM** table\_name;

## SQL JOIN

 JOIN means *to combine something*. In case of SQL, JOIN means **"to combine two or more tables"**.

The SQL JOIN clause takes records from two or more tables in a database and combines it together.

Choose columns from each table that should be used in the join. A join condition indicates a foreign key from one table

## Types of Joins

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN
* Cross Join
* Natural Join

## Different Types of SQL JOINs

**(INNER) JOIN**: Returns records that have matching values in both tables

**LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table

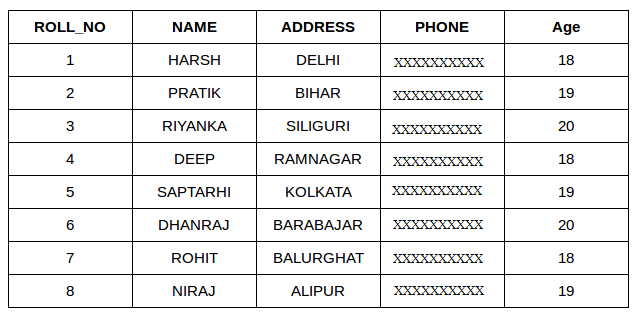
**RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table

**FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table

## INNER JOIN

The INNER JOIN keyword selects records that have matching values in both tables. **Student**

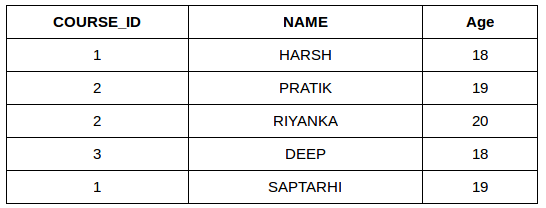
**  
 StudentCourse**

## Lightbox

**SELECT StudentCourse.COURSE\_ID, Student.NAME, Student.AGE FROM Student**

**INNER JOIN StudentCourse**

**ON Student.ROLL\_NO = StudentCourse.ROLL\_NO;**

****

## LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

LEFT JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

SELECT t1.emp\_id, t1.emp\_name, t1.hire\_date, t2.dept\_name FROM employees AS t1 LEFTJOIN departments AS t2 ON t1.dept\_id = t2.dept\_id ORDERBYemp\_id;



## RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.



SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

RIGHT JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

SELECT t1.emp\_id, t1.emp\_name, t1.hire\_date, t2.dept\_name FROM employees AS t1 RIGHTJOIN departments AS t2 ON t1.dept\_id = t2.dept\_id ORDERBYdept\_name;



## ULL OUTER JOIN

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

**Tip:** FULL OUTER JOIN and FULL JOIN are the same.

SELECT Student.NAME,StudentCourse.COURSE\_ID

FROM Student

FULL JOIN StudentCourse

ON StudentCourse.ROLL\_NO = Student.ROLL\_NO;

SELECT t1.emp\_id, t1.emp\_name, t1.hire\_date, t2.dept\_name FROM employees AS t1 FULLJOIN departments AS t2 ON t1.dept\_id = t2.dept\_id ORDERBYemp\_name;