**DBMS COMMANDS**

**DDL:** **Data Definition Language (DDL)** statements are used to define the database structure or schema.

DDL Commands:  Create, Alter, Drop, Rename, Truncate

CREATE - to create objects in the database

ALTER - alters the structure of the database

DROP - delete objects from the database

TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed

RENAME - rename an object

**DML:** **Data Manipulation Language (DML)** statements are used for managing data within schema objects and to manipulate data of a database objects.

DML Commands:   Insert, Update, Delete, Select

INSERT - insert data into a table

UPDATE - updates existing data within a table

DELETE - deletes all records from a table, the space for the records remain

SELECT - retrieve data from the a database

**3. DCL:** **Data Control Language (DCL)** statements are used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it. To control the data of a database.

DCL Commands:   Grant, Revoke

GRANT - gives user's access privileges to database

REVOKE -withdraw access privileges given with the GRANT command

**4. TCL:** **Transaction Control (TCL)** statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

TCL Commands:  Commit, Rollback, Save point

COMMIT - save work done

SAVEPOINT - identify a point in a transaction to which you can later roll back

ROLLBACK - restore database to original since the last COMMIT

**DDL (Data Definition Language) Commands: CREATE, ALTER and DROP.**

**CREATE**: This command useful for creating creating table.

Syntax:

create table [table name] (column1 datatype[size], column 2 datatype[size],… column n datatype[size] );

**Ex:**

SQL >create table student (s\_rollno number(10) primary key,s\_name varchar2(10), gender varchar2(5),dob date,addr1 varchar2(10),addr2 varchar2(10),city varchar2(10), percentage number(4));

SQL> DESC STUDENT;

Name Null? Type

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

S\_ROLLNO NOT NULL NUMBER(10)

S\_NAME VARCHAR2(10)

GENDER VARCHAR2(5)

DOB DATE

ADDR1 VARCHAR2(10)

ADDR2 VARCHAR2(10)

CITY VARCHAR2(10)

PERCENTAGE NUMBER(4)

SQL > select s\_rollno,s\_name from student;

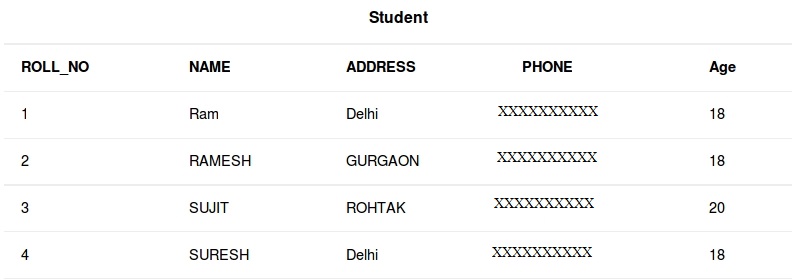
no rows selected.

**SELECT :**

Select is the most commonly used statement in SQL. The SELECT Statement in SQL is used to retrieve or fetch data from a database. We can fetch either the entire table or according to some specified rules. The data returned is stored in a result table. This result table is also called result-set.

With the SELECT clause of a SELECT command statement, we specify the columns that we want to be displayed in the query result and, optionally, which column headings we prefer to see above the result table.

The select clause is the first clause and is one of the last clauses of the select statement that the database server evaluates. The reason for this is that before we can determine what to include in the final result set, we need to know all of the possible columns that could be included in the final result set.

**Sample Table:**  
  
**Basic Syntax:**

**SELECT column1,column2 FROM table\_name**

**column1 , column2**: names of the fields of the table

**table\_name:** from where we want to fetch

This query will return all the rows in the table with fields column1 , column2.

* To fetch the entire table or all the fields in the table:

SELECT \* FROM table\_name;

* Query to fetch the fields ROLL\_NO, NAME, AGE from the table Student:

SELECT ROLL\_NO, NAME, AGE FROM Student;

Output:

| **ROLL\_NO** | **NAME** | **Age** |
| --- | --- | --- |
| 1 | Ram | 18 |
| 2 | RAMESH | 18 |
| 3 | SUJIT | 20 |
| 4 | SURESH | 18 |

* To fetch all the fields from the table Student:

SELECT \* FROM Student;

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |

# Distinct Clause

The distinct keyword is used in conjunction with select keyword. It is helpful when there is need of avoiding the duplicate values present in any specific columns/table. When we use distinct keyword only the **unique values** are fetched.

**Basic Syntax:**

**SELECT DISTINCT column1,column2 FROM table\_name**

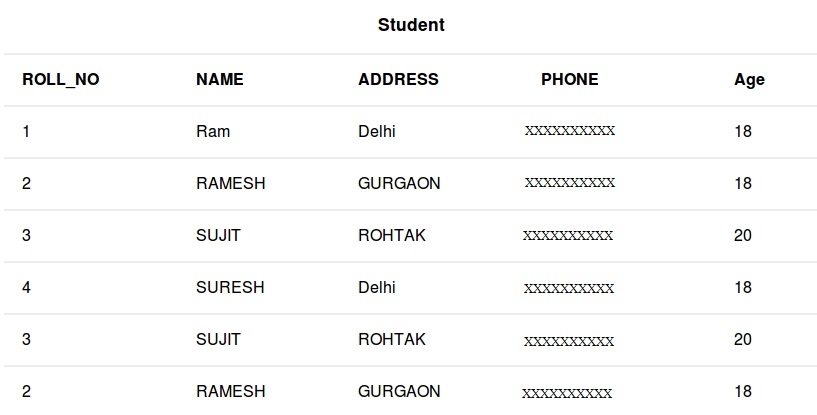
**column1 , column2:** names of the fields of the table

**table\_name:** from where we want to fetch

This query will return all the unique combination of rows in the table with fields

column1 , column2.

NOTE: If distinct keyword is used with multiple columns, the distinct combination is displayed in the result set.



To fetch unique names from the NAME field

SELECT DISTINCT NAME FROM Student;

Output:

| **NAME** |
| --- |
| Ram |
| RAMESH |
| SUJIT |
| SURESH |

To fetch unique combination of rows from the whole table

SELECT DISTINCT \* FROM Student;

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |

Please note: Without the keyword distinct in both the above examples 6 records would have been fetched instead of 4, since in the original table there are 6 records with the duplicate values.

**CREATE**

There are two CREATE statements available in SQL:

1. CREATE DATABASE
2. CREATE TABLE

**CREATE DATABASE**

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;

**CREATE TABLE**

We have learned above about creating databases. Now to store the data we need a table to do that. The CREATE TABLE statement is used to create a table in SQL. We know that a table comprises of rows and columns. So while creating tables we have to 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 particular column.

For example,**int** for integer data.

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

a column we specify the data\_type 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),

);

This query will create a table named Students. The ROLL\_NO field is of type int and can store an integer number of size 3. The next two columns NAME and SUBJECT are of type varchar and can store characters and the size 20 specifies that these two fields can hold maximum of 20 characters

**WHERE Clause :**

WHERE keyword is used for fetching **filtered data** in a result set.

* It is used to fetch data according to a particular criteria.
* WHERE keyword can also be used to filter data by matching patterns.

**Basic Syntax:**  
**SELECT column1,column2 FROM table\_name WHERE column\_name operator value;**

**column1 , column2:** fields int the table

**table\_name:** name of table

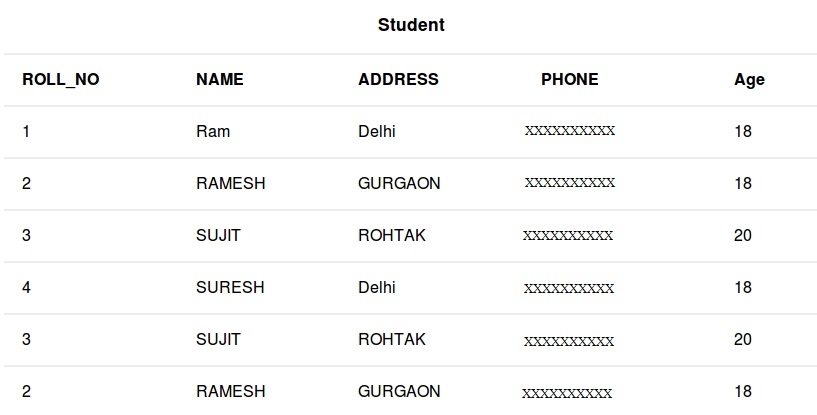
**column\_name:** name of field used for filtering the data

**operator:** operation to be considered for filtering

**value:** exact value or pattern to get related data in result

**List of operators that can be used with where clause:**

| **operator** | **description** |
| --- | --- |
| > | Greater Than |
| >= | Greater than or Equal to |
| < | Less Than |
| <= | Less than or Equal to |
| = | Equal to |
| <> | Not Equal to |
| BETWEEN | In an inclusive Range |
| LIKE | Search for a pattern |
| IN | To specify multiple possible values for a column |



**Queries**

* + To fetch record of students with age equal to 20
  + SELECT \* FROM Student WHERE Age=20;

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |

* To fetch Name and Address of students with ROLL\_NO greater than 3
* SELECT ROLL\_NO,NAME,ADDRESS FROM Student WHERE ROLL\_NO > 3;

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** |
| --- | --- | --- |
| 4 | SURESH | Delhi |

**BETWEEN** operator

It is used to fetch filtered data in a given range inclusive of two values.  
**Basic Syntax:**  
**SELECT column1,column2 FROM table\_name WHERE column\_name BETWEEN value1 AND value2;**

**BETWEEN:** operator name

**value1 AND value2:** exact value from value1 to value2 to get related data in  
result set.

**Queries**

* To fetch records of students where ROLL\_NO is between 1 and 3 (inclusive)
* SELECT \* FROM Student WHERE ROLL\_NO BETWEEN 1 AND 3;

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |

• To fetch NAME,ADDRESS of students where Age is between 20 and 30 (inclusive)

• SELECT NAME,ADDRESS FROM Student WHERE Age BETWEEN 20 AND 30;

Output:

| **NAME** | **ADDRESS** |
| --- | --- |
| SUJIT | Rohtak |
| SUJIT | Rohtak |

**LIKE operator**

It is used to fetch filtered data by searching for a particular pattern in where clause.

**Basic Syntax:**

SELECT column1,column2 FROM table\_name WHERE column\_name LIKE pattern;

LIKE: operator name

pattern: exact value extracted from the pattern to get related data in

result set.

Note: The character(s) in pattern are case sensitive.

Queries

• To fetch records of students where NAME starts with letter S.

SELECT \* FROM Student WHERE NAME LIKE 'S%';

The ‘%'(wildcard) signifies the later characters here which can be of any length and

value.More about wildcards will be discussed in the later set.

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |

• To fetch records of students where NAME contains the patter ‘AM’.

• SELECT \* FROM Student WHERE NAME LIKE '%AM%';

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |

**IN operator:**

It is used to fetch filtered data same as fetched by ‘=’ operator just the difference is that here we can specify multiple values for which we can get the result set.

Basic Syntax:

SELECT column1,column2 FROM table\_name WHERE column\_name IN (value1,value2,..);

IN: operator name

value1,value2,..: exact value matching the values given and get related data in result set.

Queries

• To fetch NAME and ADDRESS of students where Age is 18 or 20.

• SELECT NAME,ADDRESS FROM Student WHERE Age IN (18,20);

Output:

| **NAME** | **ADDRESS** |
| --- | --- |
| Ram | Delhi |
| RAMESH | GURGAON |
| SUJIT | ROHTAK |
| SURESH | Delhi |
| SUJIT | ROHTAK |
| RAMESH | GURGAON |

• To fetch records of students where ROLL\_NO is 1 or 4.

• SELECT \* FROM Student WHERE ROLL\_NO IN (1,4);

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |

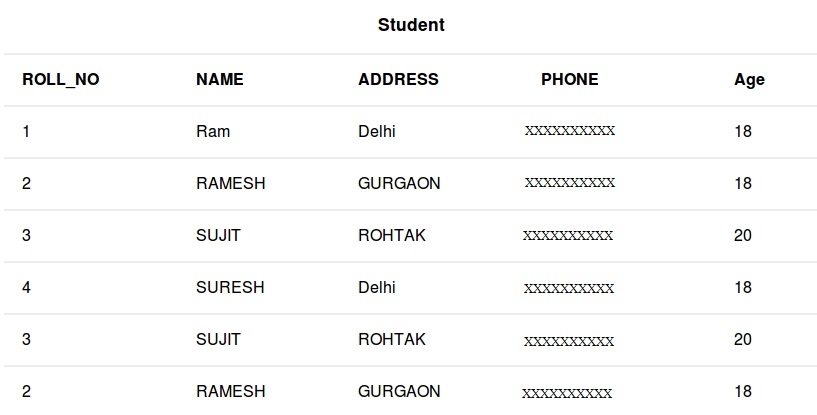
AND and OR operators

In SQL, the AND & OR operators are used for filtering the data and getting precise result based on conditions.

* The AND and OR operators are used with the WHERE clause.
* These two operators are called conjunctive operators.

**AND Operator :**This operators displays only those records where both the conditions **condition1 and condition2 evaluates to True.**

**OR Operator:**This operators displays the records where either one of the conditions condition1 and condition2 evaluates to True. That is,**either condition1 is True or condition2 is True.**



**Basic Syntax:**

**SELECT \* FROM table\_name WHERE condition1 AND condition2 and ...conditionN;**

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

**condition1,2,..N** : first condition, second condition and so on

**Sample Queries:**

1. To fetch all the records from Student table where Age is 18 and ADDRESS is Delhi.
2. SELECT \* FROM Student WHERE Age = 18 AND ADDRESS = 'Delhi';

**Output:**

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |

1. To fetch all the records from Student table where NAME is Ram and Age is 18.
2. SELECT \* FROM Student WHERE Age = 18 AND NAME = 'Ram';

**Output:**

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |

**OR Operator**

**Basic Syntax:**

**SELECT \* FROM table\_name WHERE condition1 OR condition2 OR... conditionN;**

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

**condition1,2,..N** : first condition, second condition and so on

**Sample Queries:**

1. To fetch all the records from Student table where NAME is Ram or NAME is SUJIT.
2. SELECT \* FROM Student WHERE NAME = 'Ram' OR NAME = 'SUJIT';

**Output:**

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |

1. To fetch all the records from Student table where NAME is Ram or Age is 20.
2. SELECT \* FROM Student WHERE NAME = 'Ram' OR Age = 20;

**Output:**

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |

**Combining AND and OR**

You can combine AND and OR operators in below manner to write complex queries.

Basic Syntax:

SELECT \* FROM table\_name WHERE condition1 AND (condition2 OR condition3);

Sample Queries:

1. To fetch all the records from Student table where Age is 18 NAME is Ram or RAMESH.

SELECT \* FROM Student WHERE Age = 18 AND (NAME = 'Ram' OR NAME = 'RAMESH');

Output:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |

INSERT INTO Statement

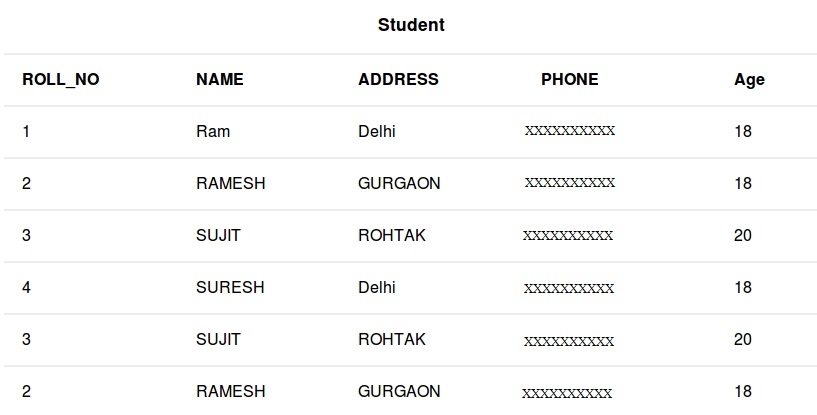
The INSERT INTO statement of SQL is used to insert a new row in a table. There are two ways of using INSERT INTO statement for inserting rows:

1. **Only values:** First method is to specify only the value of data to be inserted without the column names.

**INSERT INTO table\_name VALUES (value1, value2, value3,…);**  
**table\_name**: name of the table.  
**value1, value2,..**: value of first column, second column,… for the new record

1. **Column names and values both:** In the second method we will specify both the columns which we want to fill and their corresponding values as shown below:

**INSERT INTO table\_name (column1, column2, column3,..) VALUES ( value1, value2, value3,..);**  
**table\_name**: name of the table.  
**column1**: name of first column, second column …  
**value1, value2, value3**: value of first column, second column,… for the new record



**Method 1 (Inserting only values) :**

INSERT INTO Student VALUES (‘5′,’HARSH’,’WEST BENGAL’,’XXXXXXXXXX’,’19’);

**Output:**  
The table **Student** will now look like:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 5 | HARSH | WEST BENGAL | XXXXXXXXXX | 19 |

**Method 2 (Inserting values in only specified columns):**

INSERT INTO Student (ROLL\_NO, NAME, Age) VALUES (‘5′,’PRATIK’,’19’);

**Output:**  
The table **Student** will now look like:

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 5 | PRATIK | null | null | 19 |

Notice that the columns for which the values are not provided are filled by null. Which is the default values for those columns.

Using SELECT in INSERT INTO Statement

We can use the SELECT statement with INSERT INTO statement to copy rows from one table and insert them into another table.The use of this statement is similar to that of INSERT INTO statement. The difference is that the SELECT statement is used here to select data from a different table. The different ways of using INSERT INTO SELECT statement are shown below:

• Inserting all columns of a table: We can copy all the data of a table and insert into in a different table.

INSERT INTO first\_table SELECT \* FROM second\_table;

first\_table: name of first table.

second\_table: name of second table.

We have used the SELECT statement to copy the data from one table and INSERT INTO statement to insert in a different table.

• Inserting specific columns of a table: We can copy only those columns of a table which we want to insert into in a different table.

Syntax:

INSERT INTO first\_table(names\_of\_columns1) SELECT names\_of\_columns2 FROM second\_table;

first\_table: name of first table.

second\_table: name of second table.

names of columns1: name of columns separated by comma(,) for table 1.

names of columns2: name of columns separated by comma(,) for table 2.

We have used the SELECT statement to copy the data of the selected columns only from the second table and INSERT INTO statement to insert in first table.

• Copying specific rows from a table: We can copy specific rows from a table to insert into another table by using WHERE clause with the SELECT statement. We have to provide appropriate condition in the WHERE clause to select specific rows.

INSERT INTO table1 SELECT \* FROM table2 WHERE condition;

first\_table: name of first table.

second\_table: name of second table.

condition: condition to select specific rows.

Table2: LateralStudent

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 7 | SOUVIK | DUMDUM | XXXXXXXXXX | 18 |
| 8 | NIRAJ | NOIDA | XXXXXXXXXX | 19 |
| 9 | SOMESH | ROHTAK | XXXXXXXXXX | 20 |

**Queries:**

• Method 1(Inserting all rows and columns):

INSERT INTO Student SELECT \* FROM LateralStudent;

Output:

This query will insert all the data of the table LateralStudent in the table Student. The table Student will now look like,

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 7 | SOUVIK | DUMDUM | XXXXXXXXXX | 18 |
| 8 | NIRAJ | NOIDA | XXXXXXXXXX | 19 |
| 9 | SOMESH | ROHTAK | XXXXXXXXXX | 20 |

• Method 2(Inserting specific columns):

INSERT INTO Student(ROLL\_NO,NAME,Age) SELECT ROLL\_NO, NAME, Age FROM LateralStudent;

Output:

This query will insert the data in the columns ROLL\_NO, NAME and Age of the table LateralStudent in the table Student and the remaining columns in the Student table will be filled by null which is the default value of the remaining columns. The table Student will now look like,

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 7 | SOUVIK | null | null | 18 |
| 8 | NIRAJ | null | null | 19 |
| 9 | SOMESH | null | null | 20 |

Select specific rows to insert:

INSERT INTO Student SELECT \* FROM LateralStudent WHERE Age = 18;

Output:

This query will select only the first row from table LateralStudent to insert into the table Student. The table Student will now look like,

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 7 | SOUVIK | DUMDUM | XXXXXXXXXX | 18 |

# DELETE Statement

The DELETE Statement in SQL is used to delete existing records from a table. We can delete a single record or multiple records depending on the condition we specify in the WHERE clause.

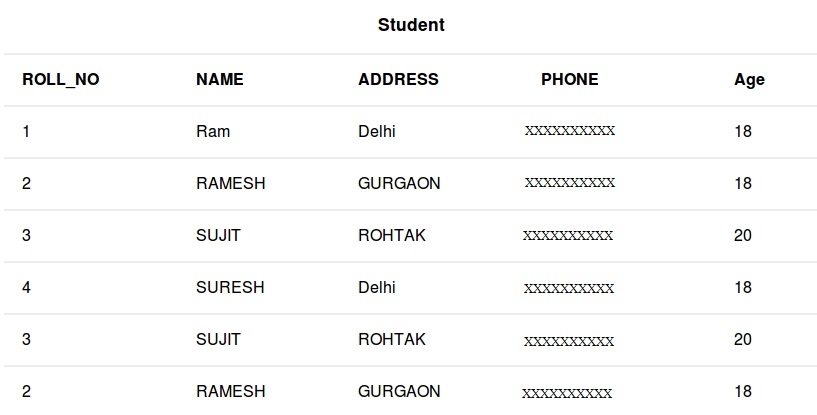
**Basic Syntax:**

DELETE FROM table\_name WHERE some\_condition;

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

**some\_condition**: condition to choose particular record.

**Note:** We can delete single as well as multiple records depending on the condition we provide in WHERE clause. If we omit the WHERE clause then all of the records will be deleted and the table will be empty.



**Example Queries:**

* **Deleting single record**: Delete the rows where NAME = ‘Ram’. This will delete only the first row.
* DELETE FROM Student WHERE NAME = 'Ram';

**Output:**  
The above query will delete only the first row and the table **Student** will now look like,

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | SUJIT | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |

* **Deleting multiple records**: Delete the rows from the table Student where Age is 20. This will delete 2 rows(third row and fifth row).
* DELETE FROM Student WHERE Age = 20;

**Output:**  
The above query will delete two rows(third row and fifth row) and the table **Student** will now look like,

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |

* **Delete all of the records:** There are two queries to do this as shown below,
* query1: "DELETE FROM Student";
* query2: "DELETE \* FROM Student";

**Output:**  
All of the records in the table will be deleted, there are no records left to display. The table **Student** will become empty!

# UPDATE Statement

The UPDATE statement in SQL is used to update the data of an existing table in database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

**Basic Syntax**

**UPDATE table\_name SET column1 = value1, column2 = value2,...**

**WHERE condition;**

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

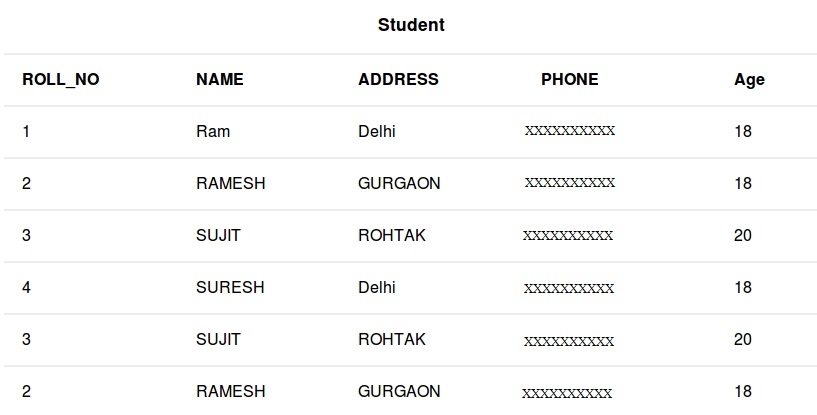
**column1**: name of first , second, third column....

**value1**: new value for first, second, third column....

**condition**: condition to select the rows for which the

values of columns needs to be updated.

**NOTE:** In the above query the**SET**statement is used to set new values to the particular column and the **WHERE** clause is used to select the rows for which the columns are needed to be updated. If we have not used the WHERE clause then the columns in **all** the rows will be updated. So the WHERE clause is used to choose the particular rows.



**Example Queries**

* + **Updating single column**: Update the column NAME and set the value to ‘PRATIK’ in all the rows where Age is 20.
  + UPDATE Student SET NAME = 'PRATIK' WHERE Age = 20;

**Output:**  
This query will update two rows(third row and fifth row) and the table **Student** will now look like,

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | Ram | Delhi | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | PRATIK | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | PRATIK | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |

* + **Updating multiple columns:** Update the columns NAME to ‘PRATIK’ and ADDRESS to ‘SIKKIM’ where ROLL\_NO is 1.
  + UPDATE Student SET NAME = 'PRATIK', ADDRESS = 'SIKKIM' WHERE ROLL\_NO = 1;

**Output**:  
The above query will update two columns in the first row and the table **Student** will now look like,

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | PRATIK | SIKKIM | XXXXXXXXXX | 18 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |
| 3 | PRATIK | ROHTAK | XXXXXXXXXX | 20 |
| 4 | SURESH | Delhi | XXXXXXXXXX | 18 |
| 3 | PRATIK | ROHTAK | XXXXXXXXXX | 20 |
| 2 | RAMESH | GURGAON | XXXXXXXXXX | 18 |

**Note:** For updating multiple columns we have used comma(,) to separate the names and values of two columns.

* **Omitting WHERE clause:** If we omit the WHERE clause from the update query then all of the rows will get updated.
* UPDATE Student SET NAME = 'PRATIK';

**Output:**  
The table **Student** will now look like,

| **ROLL\_NO** | **NAME** | **ADDRESS** | **PHONE** | **Age** |
| --- | --- | --- | --- | --- |
| 1 | PRATIK | Delhi | XXXXXXXXXX | 18 |
| 2 | PRATIK | GURGAON | XXXXXXXXXX | 18 |
| 3 | PRATIK | ROHTAK | XXXXXXXXXX | 20 |
| 4 | PRATIK | Delhi | XXXXXXXXXX | 18 |
| 3 | PRATIK | ROHTAK | XXXXXXXXXX | 20 |
| 2 | PRATIK | GURGAON | XXXXXXXXXX | 18 |

# DROP, TRUNCATE :

**DROP**

DROP is used to delete a whole database or just a table.The DROP statement destroys the objects like an existing database, table, index, or view.  
A DROP statement in SQL removes a component from a relational database management system (RDBMS).  
**Syntax:**

**DROP object object\_name**

**Examples:**

**DROP TABLE table\_name;**

**table\_name**: Name of the table to be deleted.

**DROP DATABASE database\_name;**

**database\_name**: Name of the database to be deleted.

**TRUNCATE**

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extents of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing a number of integrity enforcing mechanisms. It was officially introduced in the [SQL:2008](https://en.wikipedia.org/wiki/SQL:2008)standard.  
The TRUNCATE TABLE mytable statement is logically (though not physically) equivalent to the DELETE FROM mytable statement (without a WHERE clause).  
**Syntax:**

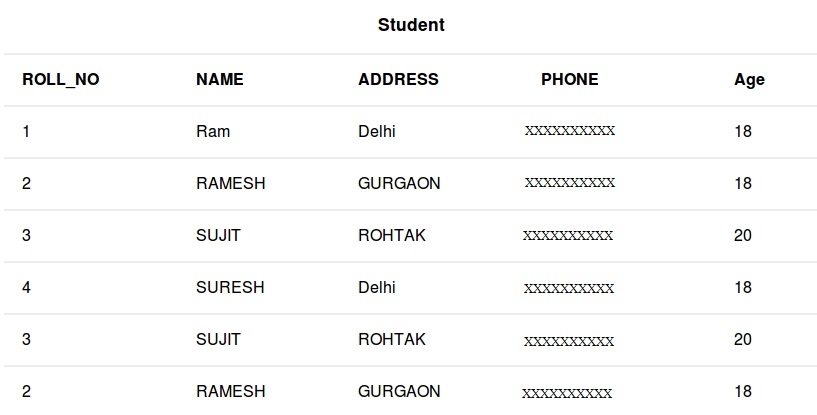
**TRUNCATE TABLE table\_name;**

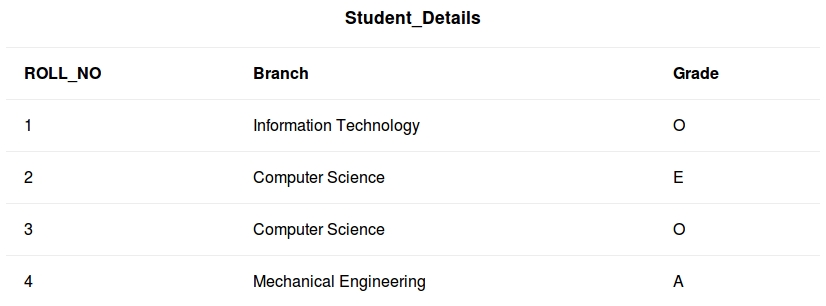
**table\_name**: Name of the table to be truncated.

**DATABASE name - student\_data**

**DROP vs TRUNCATE**

* Truncate is normally ultra-fast and its ideal for deleting data from a temporary table.
* Truncate preserves the structure of the table for future use, unlike drop table where the table is deleted with its full structure.
* Table or Database deletion using DROP statement **cannot** be rolled back, so it must be used wisely.





**Queries**

To delete the whole database

DROP DATABASE student\_data;

After running the above query whole database will be deleted.

To truncate Student\_details table from student\_data database.

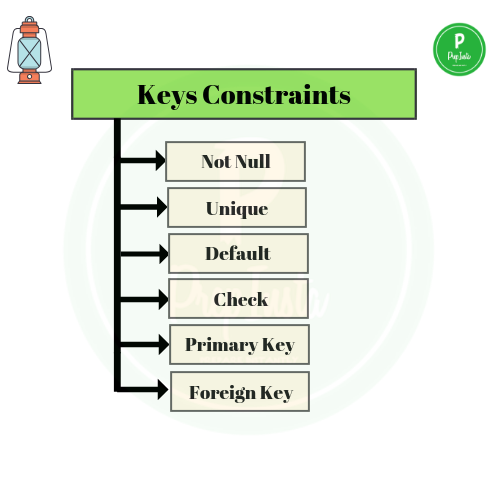
TRUNCATE TABLE Student\_details;

After running the above query Student\_details table will be truncated, i.e, the data will be deleted but the structure will remain in the memory for further operations.

# Key Constraints in DBMS

## Key Constraints in DBMS

* Constraints or nothing but the rules that are to be followed while entering data into columns of the database table
* Constraints ensure that data entered by the user into columns must be within the criteria specified by the condition
* For example, if you want to maintain only unique IDs in the employee table or if you want to enter only age under 18 in the student table etc
* We have 5 types of key constraints
  + NOT NULL: ensures that the specified **column doesn’t contain a NULL value.**
  + UNIQUE : **provides a unique/distinct values** to specified columns.
  + DEFAULT: **provides a default value** **to a column** if none is specified.
  + CHECK :**checks for the predefined conditions before inserting** the data inside the table.
  + PRIMARY KEY: it **uniquely identifies a row** in a table.
  + FOREIGN KEY: ensures **referential integrity** of the relationship

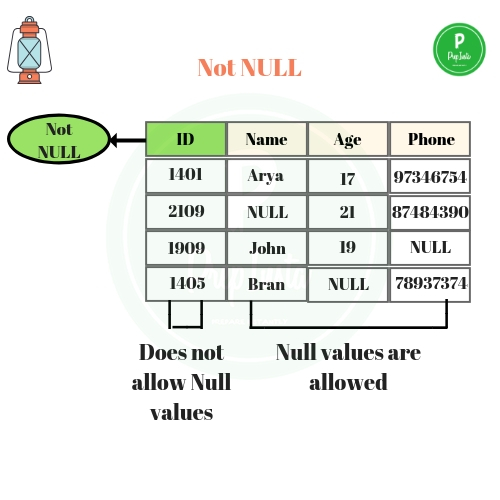


### ****Not Null****

* Null represents a record where data may be missing  data or data for that record may be optional
* Once **not null is applied to a particular column, you cannot enter null values to that column** and restricted to maintain  only some proper value other than null
* A **not-null constraint cannot be applied at table level**

#### ****Example****

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



* In the above example, we have applied not null on three columns ID, name and age which means **whenever a record is entered using insert statement all three columns should contain a value other than null**
* We have two other columns address and salary,  **where not null is not applied** which means that **you can leave the row as empty or use null value while inserting the record into the table**

### ****Unique****

* Sometimes we need to maintain only unique data   in the column of a database table, this is possible by using a unique constraint
* Unique constraint ensures that all values in a column are unique

#### ****Example****

CREATE TABLE Persons (

    ID int UNIQUE,

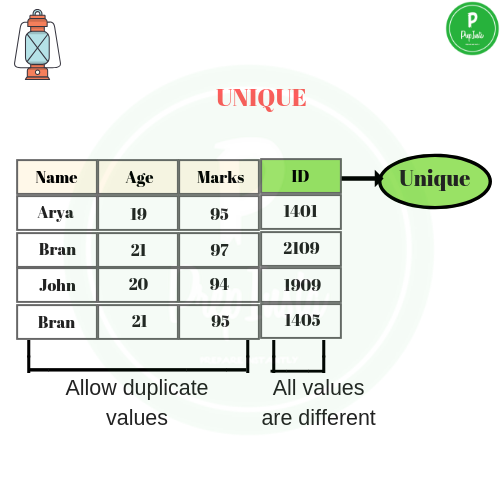
    LastName varchar(255) NOT NULL,

    FirstName varchar(255),

    Age int,

);

In the above example, **as we have used unique constraint on ID column we are not supposed to enter the data that is already present**, simply no two ID values are same

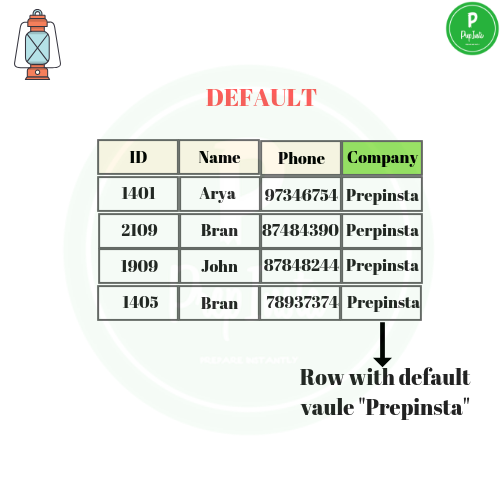


### ****DEFAULT****

* Default clause in SQL is used to add default data to the columns
* When a column is specified as default with some value then all the rows will use the same value i.e each and every time while entering the data we need not enter that value
* But **default column value can be customized** i.e it can be overridden  when inserting a data for that row based on the requirement

#### ****Example for DEFAULT clause****

The following SQL sets a DEFAULT value for the “city” column when the “emp” table is created:



#### ****My SQL / SQL Server / Oracle / MS Access:****

CREATE TABLE emp (

    ID int NOT NULL,

    LastName varchar(255) NOT NULL,

    FirstName varchar(255),

    Age int,

    City varchar(255) **DEFAULT** 'hyderabad'

);

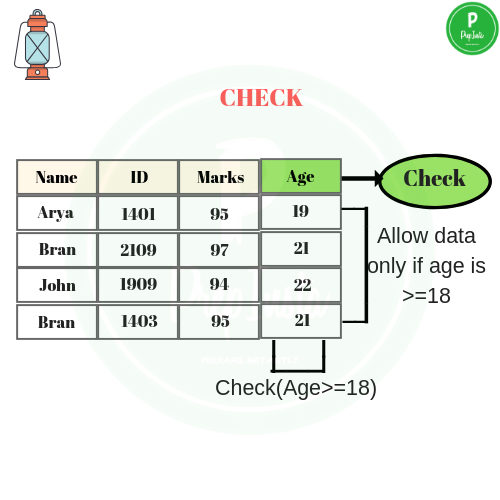
* As a result, whenever you insert a new row each time you need not enter a value for this default column  that is **entering a column value for a default column is optional and if you don’t enter the same value is considered that is used in the default clause**

### ****Check****

* Suppose in real-time if you want to give access to an application only if the age entered by the user is greater than 18 this is done at the back-end by using a check constraint
* Check constraint ensures that the data entered by the user for that column is within the range of values or possible values specified

#### ****Example for check constraint****

CREATE TABLE STUDENT (  
    ID int ,  
    Name varchar(255) ,  
    Age int,  
    CHECK (Age>=18)  
);



* As we have used a **check constraint as (Age>=18)** which means **values entered by the user for this age column while inserting the data must be less than or equal to 18** otherwise an error is shown
* Simply, the only possible values that the **age column will accept is [0 -17]**

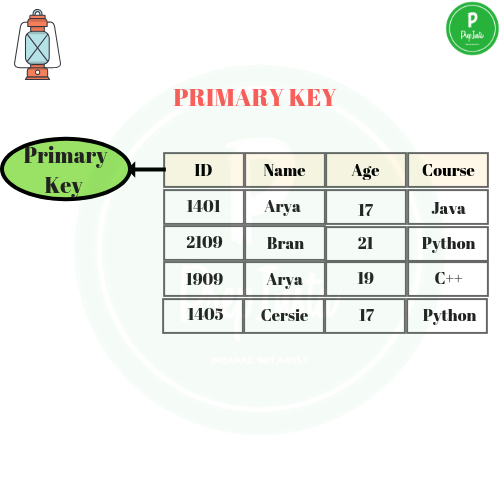
### Primary Key

* A primary key is a constraint  in a table which uniquely identifies each row record in a database table by enabling one or more the column in the table as primary key

#### Creating a primary key

A particular column is made as a primary key column by  using the primary key keyword followed with the column name

CREATE TABLE EMP (   
  ID   INT             
  NAME VARCHAR (20)         
 AGE  INT         
  COURSE VARCHAR(10)      
PRIMARY KEY (ID)   
);



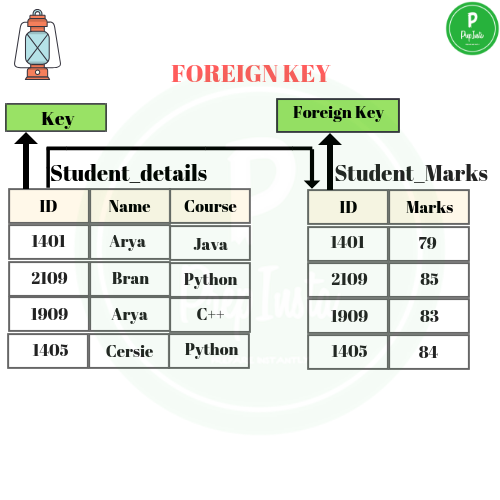
* Here we have used the primary key on ID column then ID column must contain unique values i.e **one ID cannot be used for another student**.
* If you try to **enter  duplicate value while inserting in the  row you are displayed with an error**
* Hence **primary key will restrict you to maintain unique values and not null values in that particular column**

### ****Foreign Key****

* The foreign key constraint is a column or list of columns which points to the primary key column of another table
* The main purpose of the foreign key is only those values are allowed in the present table that will match to the primary key column of another table

#### ****Example to create a foreign key****

#### ****Reference Table****



CREATE TABLE CUSTOMERS1(

   ID   INT ,

  NAME VARCHAR (20) ,  
 COURSE VARCHAR(10) ,

   PRIMARY KEY (ID)

);

#### ****Child Table****

CREATE TABLE CUSTOMERS2(

   ID   INT ,

   MARKS INT,

   REFERENCES CUSTOMERS1(ID)

);