DATATYPES

➢ NUMERICAL DATATYPES

• INTEGER DATATYPE(INT)

INT datatype is used to store integer format data.
INT datatype has been configured as

1. TINYINT
2. SMALLINT
3. MEDIUM INT
4. INT
5. BIGINT

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	**-128**	**0**	**127**	**255**
SMALLINT	2	**-32768**	**0**	**32767**	**65535**
MEDIUMINT	3	**-8388608**	**0**	**8388607**	**16777215**
INT	4	**-2147483648**	**0**	**2147483647**	**4294967295**
BIGINT	8	**-2^{63}**	**0**	**2^{63}-1**	**2^{64}-1**

• DECIMAL DATATYPE

The DECIMAL or NUMERIC types store exact numeric data values or decimal values. These types are used when it is important to preserve exact precision.

DECIMAL datatype takes precision and scale as a argument.
Syntax: `DECIMAL(precision [,scale]);`

Case 1: When only precision is passed,

DECIMAL(3);

± | 9 | 9 | 9 |

Case 2: When precision is more than scale i.e. (P>=S),

DECIMAL(6,4);

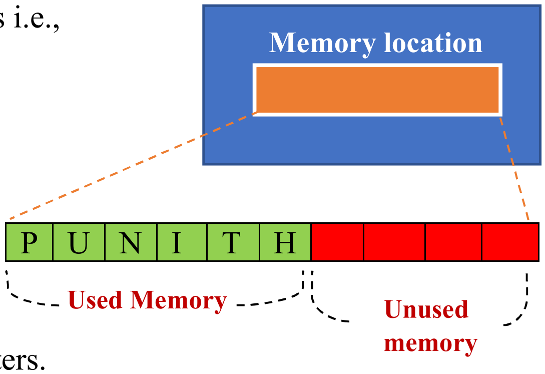
± | 9 | 9 | 9. | 9 | 9 | 9 |

DECIMAL(6,6);

± **0.** | 9 | 9 | 9 | 9 | 9 | 9 |

CHARACTER DATATYPES

- **CHAR:** It is used to store the characters i.e.,
  - UPPER CASE
  - lower case
  - Special Characters
  - Numbers '0-9'

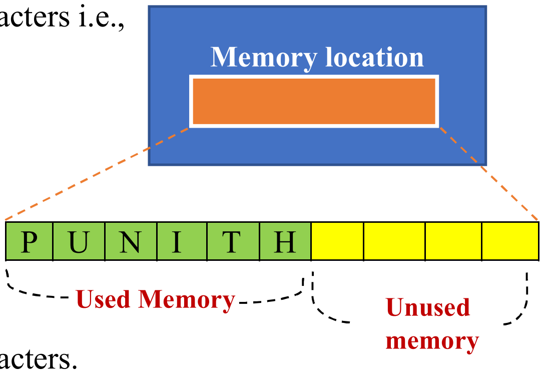


Syntax: CHAR(**size**)

Ex: CHAR(10)

- CHAR can store up to 255 characters.
- In case of CHAR, the unused memory cannot be given back to memory location for re-use purpose.
- Hence there is a wastage of memory.
- CHAR datatype is also known as **'Fixed Length Memory Location'**.

- **VARCHAR:** It is used to store the characters i.e.,
  - UPPER CASE
  - lower case
  - Special Characters
  - Numbers '0-9'



Syntax: CHAR(**size**)

Ex: VARCHAR(10)

- CHAR can store up to 65535 characters.
- In case of CHAR, the unused memory can be given back to memory location for re-use purpose.
- Hence there is no wastage of memory.
- VARCHAR datatype is also known as **'Variable Length Memory Location'**.

- **LARGE OBJECTS**

  - **CHARACTER LARGE OBJECT(CLOB)**
    - It is used to store the large amount of characters.
    - We can store up to 4GB.

Syntax: CLOB;

- **BINARY LARGE OBJECT(BLOB)**
  - It is used to store the binary object such as audio files, video files, doc files, image files etc.

Syntax: BLOB

- **CHARACTER DATATYPES**

- **DATE:** It is used to store the date format data.
  - **Standard Date Format**
    DD/MM/YY → 01/06/23
    DD/MM/YYYY → 01/06/2023
  - **Oracle Date Format**
    DD-MON-YY → 01-JUN-23
    DD-MON-YYYY → 01-JUN-2023
  - **MySQL Date Format**
    YYYY-MM-DD → 2023-06-01

- **DATETIME:** It is used to store both date and time.
  Format: YYYY-MM-DD hh:mi:ss

- **TIMESTAMP:** It is used to store date, time and the current time one.
  Format: YYYY-MM-DD hh:mi:ss UTC.

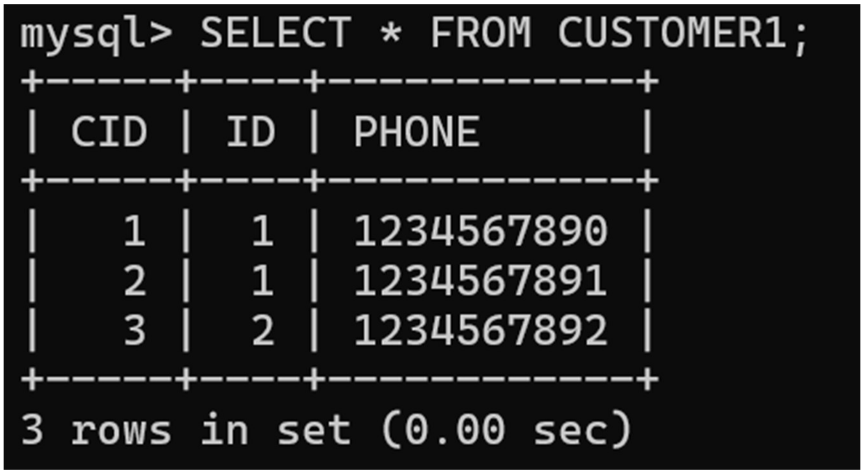
CONSTRAINTS

They are the rules/constraints given to the columns to validate the data.

Types of Constraints:

- UNIQUE
- NOT NULL
- CHECK
- PRIMARY KEY
- FOREIGN KEY

CUSTOMER1 TABLE



```
mysql> SELECT * FROM CUSTOMER1;
+-----+----+------------+
| CID | ID | PHONE      |
+-----+----+------------+
|   1 |  1 | 1234567890 |
|   2 |  1 | 1234567891 |
|   3 |  2 | 1234567892 |
+-----+----+------------+
3 rows in set (0.00 sec)
```

UNIQUE: This constraint is used to avoid duplicate values which are entered in the table.

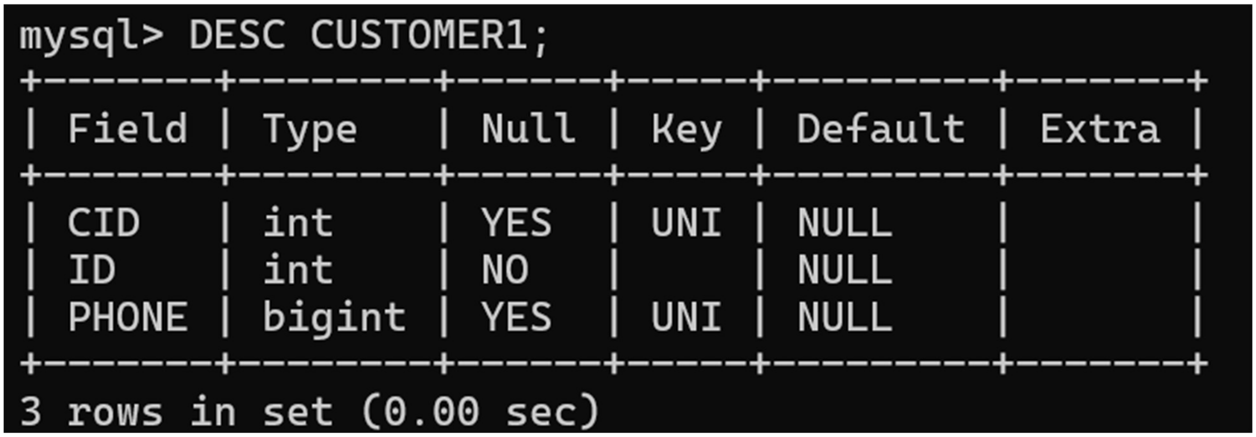
NULL

- It is a keyword which represents empty/nothing.
- NULL doesn’t occupy any memory.
- Any operation performed with null, the resultant will be null.
- Two nulls are not same in RDBMS.

NOT NULL: This constraint is used to represent the cell should not be empty or This constraint is used to represent the non-empty cell.

CHECK(condition): This constraint is used to provide user defined condition or customized condition based on requirement.

Ex: CHECK(LENGTH(CONTACT)=10);
CHECK(PRICE)>100;
CHECK(PERC>=60&&PERC<=100);



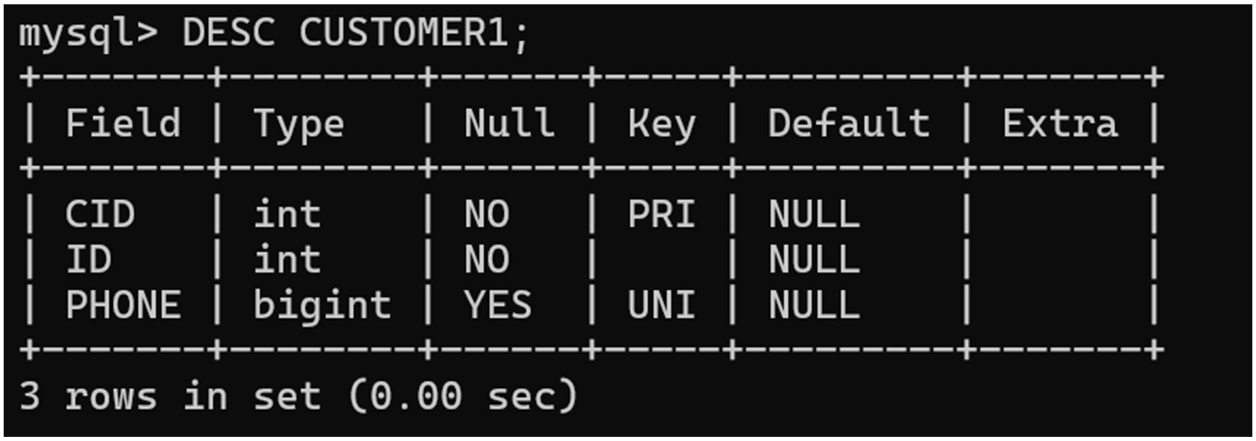
```
mysql> DESC CUSTOMER1;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| CID   | int    | YES  | UNI | NULL    |       |
| ID    | int    | NO   |     | NULL    |       |
| PHONE | bigint | YES  | UNI | NULL    |       |
+-------+--------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

PRIMARY KEY

Primary Key is used to uniquely identify the records from the table.

Characteristics of PRIMARY Key

- Used to represent the table among the schema of UMIQUE and NOTNULL.
- Must be UNIQUE and NOTNULL.
- As per the standards, a table can have only one PRIMARY key.
- There might be many UNIQUE constraints defined per table, but only one primary key defined per table.



```
mysql> DESC CUSTOMER1;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| CID   | int    | NO   | PRI | NULL    |       |
| ID    | int    | NO   |     | NULL    |       |
| PHONE | bigint | YES  | UNI | NULL    |       |
+-------+--------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

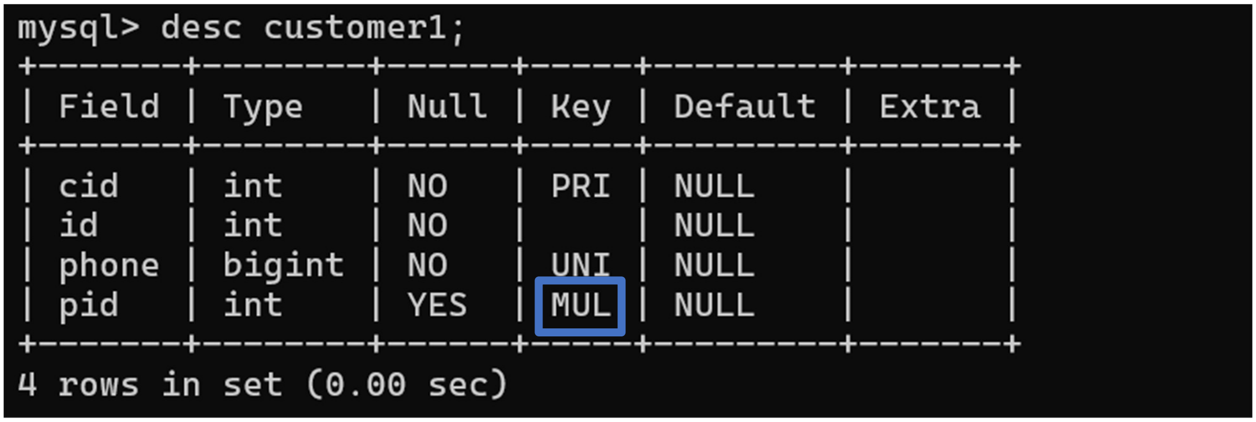
FOREIGN KEY

Foreign Key is used to establish the connection between multiple tables.

Characteristics of FOREIGN Key

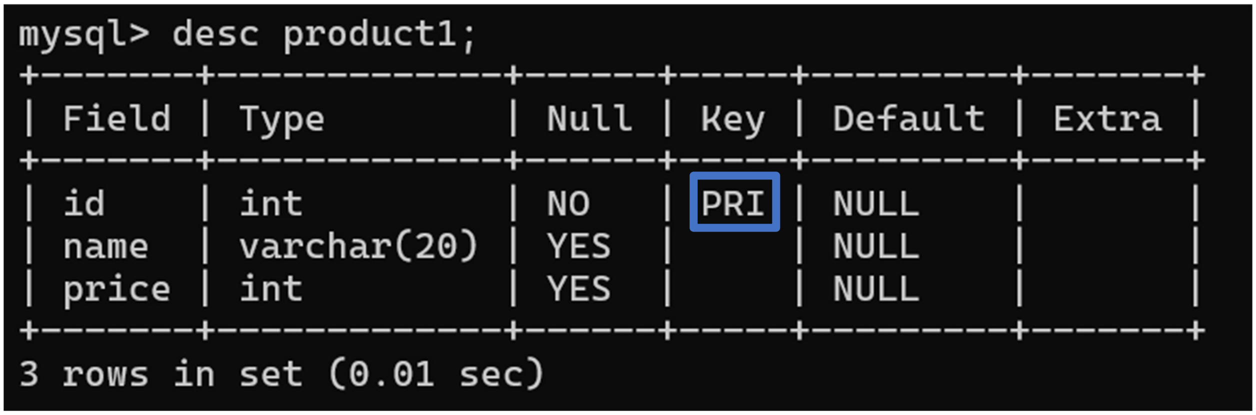
- Table can have multiple foreign key.
- PRIMARY Key of a table can be eligible to become FOREIGN Key of another table.
- FOREIGN Key can accept the duplicate values and also NULL Values, but it is not mandatory.
- FOREIGN Key present in Child Table but always belongs to Parent table.
- FOREIGN KEY is also known as **Referential Integrity Constraints.**

CUSTOMER1 TABLE



```
mysql> desc customer1;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| cid   | int    | NO   | PRI | NULL    |       |
| id    | int    | NO   |     | NULL    |       |
| phone | bigint | NO   | UNI | NULL    |       |
| pid   | int    | YES  | MUL | NULL    |       |
+-------+--------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

PRODUCT1 TABLE



```
mysql> desc product1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | NO   | PRI | NULL    |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| price | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

DEFAULT : This constraint is used to set a default value for a particular column in the table.
Syntax: column_name DATATYPE **DEFAULT 'default_value';**

AUTO_INCREMENT: Auto increment is used to generate a unique value automatically for a column in a table.
Syntax: column_name DATATYPE **auto_increment[=value];**

ENUM: This constraint acts like a datatype and used to set only limited set of values for a particular column.
Syntax: column_name **ENUM(v1, v2, .., vN);**