Insert Ignore

In MySQL, the INSERT INTO statement can be used to insert one or more records into a table.

In some scenarios, if a particular column has a UNIQUE constraint and if we are trying to add duplicates records into that particular column using the INSERT INTO statement, MySQL will terminate the statement and returns an error. As the result, no rows are inserted into the table.

MySQL Insert Ignore Statement

However, if we use the MySQL **INSERT IGNORE INTO** statement, it will not display an error. Instead, it allows us to insert valid data into a table and ignores the rows with invalid data that would cause errors.

Following are some scenarios where the INSERT IGNORE INTO statement avoid errors:

- When we insert a duplicate value in the column of a table that has UNIQUE key or PRIMARY key constraints.
- When we try to add NULL values to a column where it has NOT NULL constraint on it.

Syntax

Following is the syntax of the INSERT IGNORE statement in MySQL –

```
 INSERT IGNORE INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Example

First of all, let us create a table named CUSTOMERS using the following query below –

Note: The UNIQUE constraint ensures that no duplicate value can be stored or inserted in the NAME column.

```
CREATE TABLE CUSTOMERS (
  ID int NOT NULL,
  NAME varchar(20) NOT NULL UNIQUE,
  PRIMARY KEY (ID)
);
INSERT INTO CUSTOMERS (ID, NAME)
VALUES (1, "Ajay"), (2, "Vinay"), (3, "Arjun");

SELECT * FROM CUSTOMERS;
```

Now, let us insert a duplicate record into the NAME column of CUSTOMERS table using the below query –

```
INSERT INTO CUSTOMERS (NAME) VALUES (2, "Arjun");
```

```
ERROR 1062 (23000): Duplicate entry 'Arjun' for key 'customers.NAME'
```

```
INSERT IGNORE INTO CUSTOMERS (NAME) VALUES (2, "Arjun");
```

We can find the details of the above warning using the following query

```
SHOW WARNINGS;
```

```
SELECT * FROM CUSTOMERS;
```

MySQL INSERT IGNORE and STRICT mode

The strict mode controls how MySQL handles the invalid, missing, or out of range values that are going to be added into a table through data-change statements such as **INSERT** or **UPDATE**.

So, if the strict mode is **ON**, and we are trying to insert some invalid values into a table using the INSERT statement, MySQL terminates the statement returns an error message.

However, if we use the INSERT IGNORE INTO statement, instead of returning an error, MySQL will adjust those values to make them valid before adding the value to the table.

Example

Let us create a table named CUSTOMERS using the following query −

Note: The NAME column accepts only strings whose length is less than or equal to five.

```
CREATE TABLE CUSTOMERS (
  ID int NOT NULL,
  NAME varchar(5),
  PRIMARY KEY (ID)
);
```

```
INSERT INTO CUSTOMERS (NAME) VALUES (1, "Malinga");
```

```
ERROR 1406 (22001): Data too long for column 'NAME' at row 1
```

```
INSERT IGNORE INTO CUSTOMERS (NAME) VALUES (1, "Malinga");
```

```
SHOW WARNINGS;
```

```
Select * from CUSTOMERS;
```

Insert Ignore Query Using a Client Program

```
insert_ignore_query = "INSERT IGNORE INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...)"
cursorObj.execute(insert_ignore_query)
```

```
import mysql.connector
#establishing the connection
connection = mysql.connector.connect(
    host='localhost',
    user='root',
    password='password',
    database='tut'
)
cursorObj = connection.cursor()
insert_ignore_query = "INSERT IGNORE INTO tutorials_tbl (tutorial_id, tutorial_title, tutorial_author, submission_date) VALUES (7, 'New Tutorial', 'John Doe', '2023-07-25')"
cursorObj.execute(insert_ignore_query)
connection.commit()
print("INSERT IGNORE query executed successfully.")
cursorObj.close()
connection.close()
```