

## Study Unit 4: MySQL (Constraints, Not Null, Unique, Primary and Foreign key etc.)

### Outline

- Introduction to MySQL Constraints
- SQL Injection
- SQL Hosting

### Learning Outcomes of Study Unit 4

Upon completion of this study unit, you should be able to

- 1.1 MySQL Constraints
  - Example of Constraint
  - Not Null
  - Unique
- 1.2 Primary key Constraints
  - Primary Key on Create Table
  - Primary Key on Alter Table
  - Drop a Primary Key
- 1.3 Foreign Key Constraints
  - Foreign Key on Create Table
  - Foreign Key on Alter Table
  - Drop a Foreign Key
- 1.4 SQL Injection
  - Preventing SQL Injection
- 1.5 Most Common SQL Hosting Databases
  - MS SQL Server
  - Oracle
  - MySQL
  - MS Access

## 1.1 MySQL Constraints

SQL constraints are used to specify rules for data in a table.

SQL Create Constraints

Constraints can be specified when the table is created with the **CREATE TABLE** statement, or after the table is created with the **ALTER TABLE** statement

### Syntax

```
CREATE TABLE table_name (  
    column1 datatype constraint,  
    column2 datatype constraint,  
    column3 datatype constraint,  
    ....  
);
```

### 1.1.1 Example of MySQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.


The following constraints are commonly used in SQL:



- [NOT NULL](#) - Ensures that a column cannot have a NULL value
- [UNIQUE](#) - Ensures that all values in a column are different
- [PRIMARY KEY](#) - A combination of a **NOT NULL** and **UNIQUE**. Uniquely identifies each row in a table
- [FOREIGN KEY](#) - Prevents actions that would destroy links between tables etc....

### Example:

```
CREATE TABLE CONSTRAINT_Exp (
    id int PRIMARY KEY,
    Name varchar(255) NOT null,
    email varchar(255)
);
```

### OUTPUT:

	#	Name	Type	Collation	At
<input type="checkbox"/>	1	id 	int(11)		
<input type="checkbox"/>	2	Name	varchar(255)	utf8mb4_general_ci	
<input type="checkbox"/>	3	email	varchar(255)	utf8mb4_general_ci	

☐ Check all    With selected:  Browse  
 Remove from central columns

### 1.1.2 Not Null

By default, a column can hold NULL values.

The **NOT NULL** constraint enforces a column to NOT accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field

#### SQL NOT NULL on CREATE TABLE

The following SQL ensures that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:

```
CREATE TABLE Persons2 (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

The following query will execute but FirstName record will empty.

It is because of there is no constraint of (Not Null) by default null values can be insert in FirstName

```
INSERT INTO `persons2`(`ID`,`LastName`,`Age`) VALUES (1,'saleem',28);
```

+ Options

ID	LastName	FirstName	Age
1	saleem	NULL	28

If we skip LastName field than you will find a warning that this field doesn't have any Default Value.

Note: Default value is NULL

✓ 1 row inserted. (Query took 0.0664 seconds.)

```
INSERT INTO `persons2`(`ID`,`FirstName`,`Age`) VALUES (1,'sajid',28)
```

⚠ Warning: #1364 Field 'LastName' doesn't have a default value

## SQL NOT NULL on ALTER TABLE

To create a **NOT NULL** constraint on the "Age" column when the "Persons" table is already created, use the following SQL:

```
ALTER TABLE Persons
MODIFY Age int NOT NULL;
```

### 1.1.3 Unique Constraints

The **UNIQUE** constraint ensures that all values in a column are different.

Both the **UNIQUE** and **PRIMARY KEY** constraints provide a guarantee for uniqueness for a column or set of columns.

A **PRIMARY KEY** constraint automatically has a **UNIQUE** constraint.

However, you can have many **UNIQUE** constraints per table, but only one **PRIMARY KEY** constraint per table.

**MySQL:**

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

#	Name	Type	Collation	Attributes	N
<input type="checkbox"/> 1	<b>ID</b>	int(11)			N
<input type="checkbox"/> 2	<b>LastName</b>	varchar(255)	utf8mb4_general_ci		N
<input type="checkbox"/> 3	<b>FirstName</b>	varchar(255)	utf8mb4_general_ci		Y
<input type="checkbox"/> 4	<b>Age</b>	int(11)			Y

☐ Check all    With selected:

Inserting values into the table with table id = 1(id is unique)

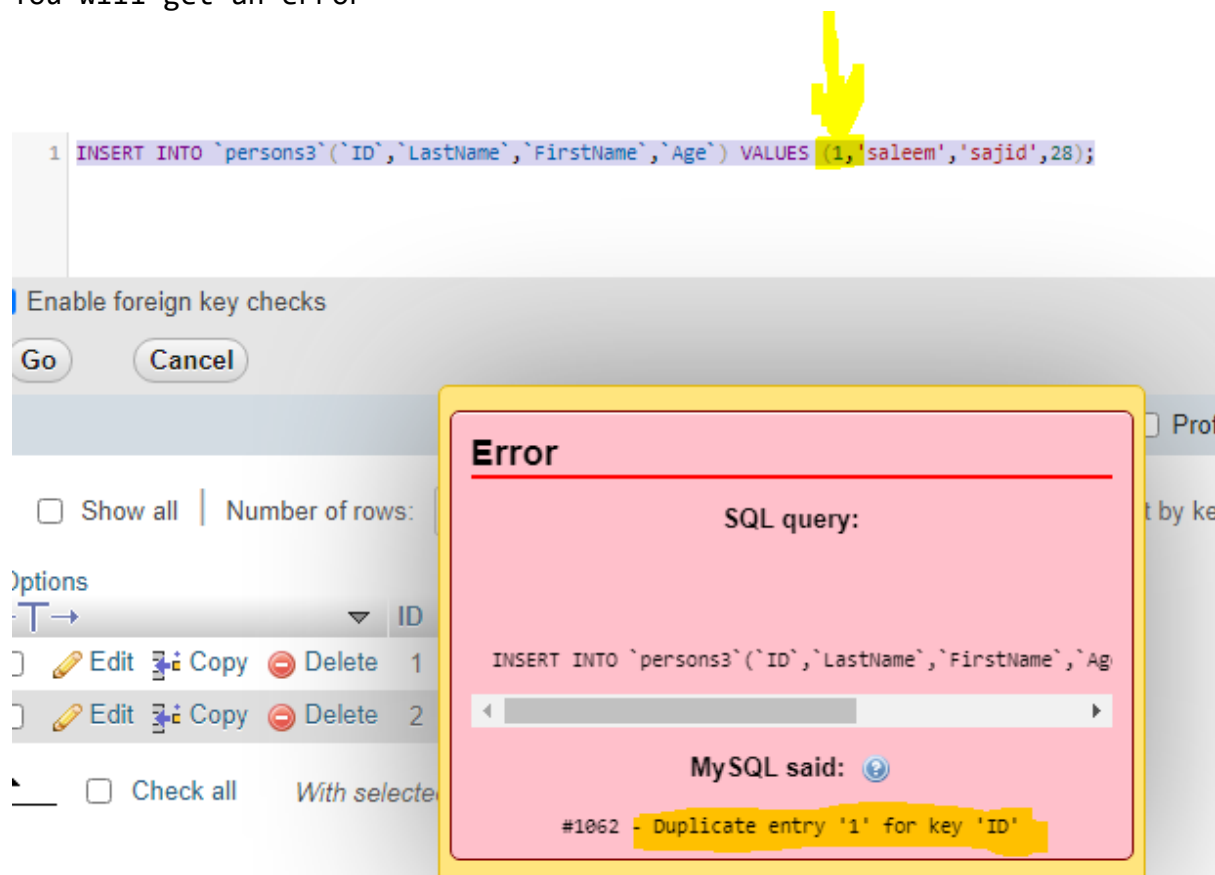
```
INSERT INTO `persons3` (`ID`,`LastName`,`FirstName`,`Age`) VALUES
(1,'saleem','sajid',28);
```

+ Options

	ID	LastName	FirstName	Age
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	saleem	sajid	28
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	saleem	sajid	28

☐ Check all    With selected:

If you will try to insert data with id=1.  
You will get an error



1 `INSERT INTO `persons3` (`ID`, `LastName`, `FirstName`, `Age`) VALUES (1, 'saleem', 'sajid', 28);`

Enable foreign key checks

Go Cancel

☐ Show all | Number of rows:

Options

⌵ ID

⌵ Edit Copy Delete 1

⌵ Edit Copy Delete 2

☐ Check all With selecte

**Error**

SQL query:

`INSERT INTO `persons3` (`ID`, `LastName`, `FirstName`, `Age``

MySQL said: ⓘ

#1062 - Duplicate entry '1' for key 'ID'

To name a **UNIQUE** constraint, and to define a **UNIQUE** constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT UC_Person UNIQUE (ID, LastName)  
);
```

### 1.1.4 Unique Constraints on Alter Table

To create a **UNIQUE** constraint on the "ID" column when the table is already created, use the following SQL:

#### MySQL:

```
ALTER TABLE Persons3  
ADD UNIQUE (ID);
```

### 1.1.5 Drop a Unique Constraints

To drop a **UNIQUE** constraint, use the following SQL:

#### MySQL:

```
ALTER TABLE Persons3  
DROP INDEX ID;
```

## 1.2 Primary Key Constraints

The **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; and in the table, this primary key can consist of single or multiple columns (fields).

### 1.2.1 Primary Key on Create Table

The following SQL creates a **PRIMARY KEY** on the "ID" column when the "Persons" table is created:

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

OUTPUT:

	#	Name	Type	Collation	Attribute
<input type="checkbox"/>	1	ID 	int(11)		
<input type="checkbox"/>	2	LastName	varchar(255)	utf8mb4_general_ci	
<input type="checkbox"/>	3	FirstName	varchar(255)	utf8mb4_general_ci	
<input type="checkbox"/>	4	Age	int(11)		

### 1.2.2 Primary Key on Alter Table

To create a **PRIMARY KEY** constraint on the "ID" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

### 1.2.3 Drop Primary Key Constraints

To drop a **PRIMARY KEY** constraint, use the following SQL:

**MySQL:**

```
ALTER TABLE Persons
DROP PRIMARY KEY;
```

## 1.3 Foreign Key Constraints

The **FOREIGN KEY** constraint is used to prevent actions that would destroy links between tables.

A **FOREIGN KEY** is a field (or collection of fields) in one table, that refers to the **PRIMARY KEY** in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Look at the following two tables:

## Persons Table

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

## Orders Table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2

4	24562	1
---	-------	---

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the **PRIMARY KEY** in the "Persons" table.


The "PersonID" column in the "Orders" table is a **FOREIGN KEY** in the "Orders" table.

The **FOREIGN KEY** constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

### 1.3.1 Foreign Key on Create Table

The following SQL creates a **FOREIGN KEY** on the "PersonID" column when the "Orders" table is created:

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (PersonID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

	#	Name	Type	Collation	Attribute
<input type="checkbox"/>	1	OrderID	int(11)		
<input type="checkbox"/>	2	OrderNumber	int(11)		
<input type="checkbox"/>	3	PersonID 	int(11)		

### 1.3.2 Foreign Key on Alter Table

The following SQL creates a **FOREIGN KEY** on the "PersonID" column when the "Orders" table is created:

```
ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

### 1.3.3 Drop a Foreign Key Constraints

To drop a **FOREIGN KEY** constraint, use the following SQL:

**MySQL:**

```
ALTER TABLE Orders  
DROP FOREIGN KEY PersonID;
```

## 1.4 SQL Injection

- SQL injection is a code injection technique that might destroy your database.
- SQL injection is one of the most common web hacking techniques.
- SQL injection is the placement of malicious code in SQL statements, via web page input.

## Use SQL Parameters for Protection

To protect a web site from SQL injection, you can use SQL parameters.

SQL parameters are values that are added to an SQL query at execution time, in a controlled manner.

If you take user input through a webpage and insert it into a MySQL database, there's a chance that you have left yourself wide open for a security issue known as **SQL Injection**. This chapter will teach you how to help prevent this from happening and help you secure your scripts and MySQL statements.

The SQL Injection usually occurs when you ask a user for input, like their name and instead of a name they give you a MySQL statement that you will unknowingly run on your database.

Never trust the data provided by a user, process this data only after validation; as a rule, this is done by pattern matching. In the following example, the username is restricted to alphanumerical characters plus underscore and to a length between 8 and 20 characters – modify these rules as needed.

```
if (preg_match("/^\w{8,20}$/", $_GET['username'], $matches)) {  
    $result = mysql_query("SELECT * FROM users WHERE username = $matches[0]");  
} else {  
    echo "username not accepted";  
}
```

To demonstrate this problem, consider the following excerpt.

```
// supposed input  
$name = "Qadir'; DELETE FROM users;";  
mysql_query("SELECT * FROM users WHERE name = '{$name}'");
```

The function call is supposed to retrieve a record from the users table, where the name column matches the name specified by the user. Under normal circumstances, \$name would only contain alphanumeric characters and perhaps spaces. But here, by appending an entirely new query to \$name, the call to the database turns into a disaster. The injected DELETE query removes all the records from users.

Fortunately, if you use MySQL, the `mysql_query()` function does not permit query stacking or executing multiple queries in a single function call. If you try to stack queries, the call fails.

However, other PHP database extensions, such as **SQLite** and **PostgreSQL**, happily perform stacked queries, executing all the queries provided in one string and creating a serious security problem.

### 1.4.1 Preventing SQL Injection

You can handle all escape characters smartly in scripting languages like PERL and PHP. The MySQL extension for PHP provides the function **mysql\_real\_escape\_string()** to escape input characters that are special to MySQL.

```
$name = mysql_real_escape_string($name);  
mysql_query("SELECT * FROM users WHERE name = '{$name}'");
```

## 1.5 Most Common SQL Hosting Databases

If you want your web site to be able to store and retrieve data from a database, your web server should have access to a database-system that uses the SQL language.

If your web server is hosted by an Internet Service Provider (ISP), you will have to look for SQL hosting plans. The most common SQL hosting databases are MS SQL Server, Oracle, MySQL, and MS Access.

### 1.5.1: MS SQL Server

Microsoft's SQL Server is a popular database software for database-driven web sites with high traffic.

SQL Server is a very powerful, robust and full featured SQL database system.

### 1.5.2: Oracle

Oracle is also a popular database software for database-driven web sites with high traffic.

Oracle is a very powerful, robust and full featured SQL database system.

### 1.5.3: MySQL

MySQL is also a popular database software for web sites.

MySQL is a very powerful, robust and full featured SQL database system.

MySQL is an inexpensive alternative to the expensive Microsoft and Oracle solutions.

### 1.5.4: MS Access

When a web site requires only a simple database, Microsoft Access can be a solution.

MS Access is not well suited for very high-traffic, and not as powerful as MySQL, SQL Server, or Oracle.

## **Self-Review Questions (SRQ) For Study Session 4**

Now that you have completed this study unit, you can assess how well you have achieved its Learning Outcomes by answering these questions. Write your answers in your Study Diary and discuss them with your Tutor at the next Study Support Meeting or Online interactive sessions.

1. What are SQL Constraints and Syntax? Which constraints are commonly used in SQL?
2. Explain SQL Unique and Not NULL Constraints?
3. Why we use Primary Key in a Table? Give Examples
4. How to create Primary Key to (Persons) table on the (ID) column.
5. How To drop a PRIMARY KEY constraint, use the MySQL statement?
6. Create a FOREIGN KEY on the "Person ID" column when the "Orders" table.
7. Describe about MySQL injection in detail.
8. What is SQL Hosting? What are the most common SQL hosting databases?

## References and Additional Reading Materials

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/mysql/index.htm>