Week 10

MySQL Constraints

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

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,
);
```

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.

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:

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT

MySQL NOT NULL Constraint

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.

NOT NULL on CREATE TABLE

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

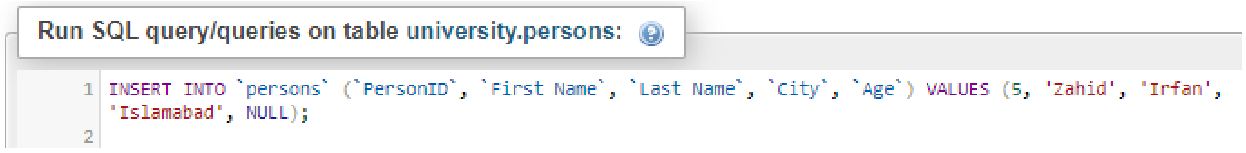
```
CREATE TABLE Persons (
   PersonID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255) NOT NULL,
   Age int
);
```

NOT NULL on ALTER TABLE

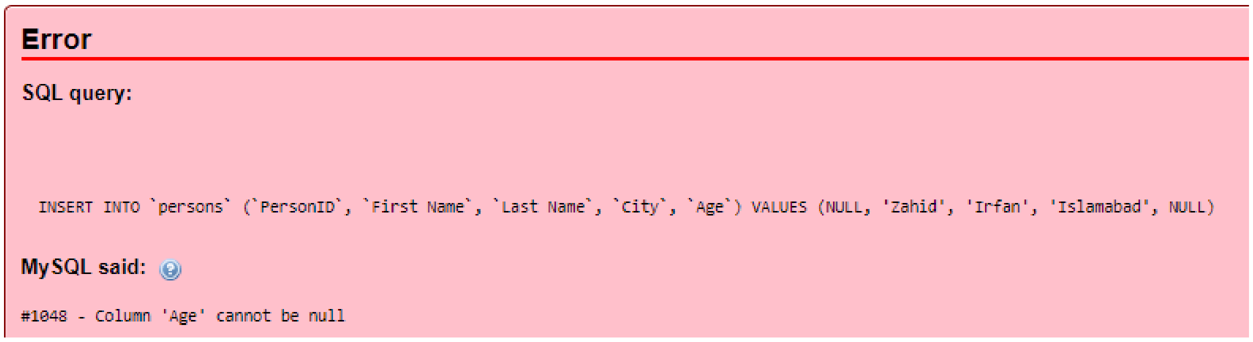
To create a NOT NULL constraint on the "Age" column when the "Persons" table is already created, use the following SQL:

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

Example:



```
INSERT INTO `persons` (`PersonID`, `First Name`, `Last Name`, `City`, `Age`) VALUES (5, 'Zahid', 'Irfan',
'Islamabad', NULL);
```


Error

SQL query:

```
INSERT INTO `persons` (`PersonID`, `First Name`, `Last Name`, `City`, `Age`) VALUES (NULL, 'Zahid', 'Irfan', 'Islamabad', NULL)
```

MySQL said:

```
#1048 - Column 'Age' cannot be null
```

MySQL UNIQUE Constraint

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.

UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "PersonID" column when the "Persons" table is created:

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

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

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

UNIQUE Constraint on ALTER TABLE

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

```
ALTER TABLE Persons
ADD UNIQUE (FirstName);
```

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

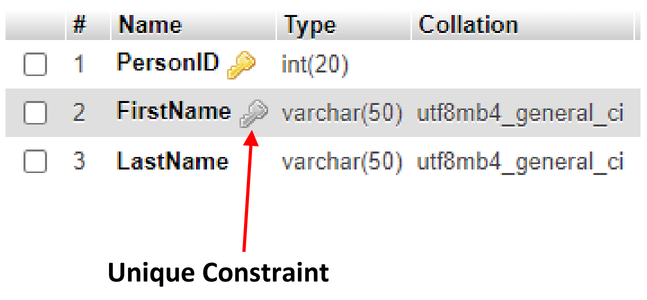
```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (PersonID,FirstName);
```

DROP a UNIQUE Constraint

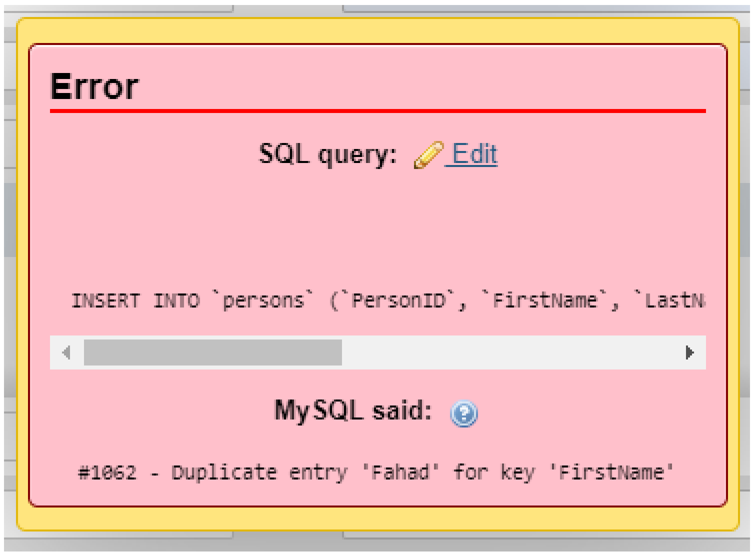
To drop a UNIQUE constraint, use the following SQL:

```
ALTER TABLE Persons
DROP INDEX UC_Person;
```

Example:



	#	Name	Type	Collation
☐	1	PersonID	int(20)	
☐	2	FirstName	varchar(50)	utf8mb4_general_ci
☐	3	LastName	varchar(50)	utf8mb4_general_ci



```
INSERT INTO `persons` (`PersonID`, `FirstName`, `LastN
```

```
#1062 - Duplicate entry 'Fahad' for key 'FirstName'
```

MySQL PRIMARY KEY Constraint

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).

PRIMARY KEY on CREATE TABLE

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

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

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

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

Note: In the example above there is only ONE PRIMARY KEY (PK_Person). However, the VALUE of the primary key is made up of TWO COLUMNS (PersonID + LastName).

PRIMARY KEY on ALTER TABLE

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

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

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

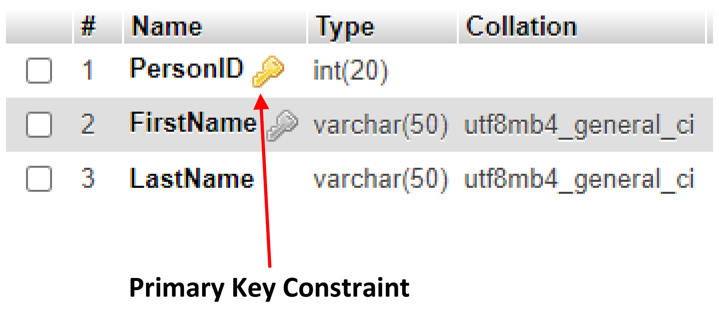
```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (PersonID,LastName);
```

DROP a PRIMARY KEY Constraint

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

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

Example:



	#	Name	Type	Collation
☐	1	PersonID	int(20)	
☐	2	FirstName	varchar(50)	utf8mb4_general_ci
☐	3	LastName	varchar(50)	utf8mb4_general_ci

MySQL FOREIGN KEY Constraint

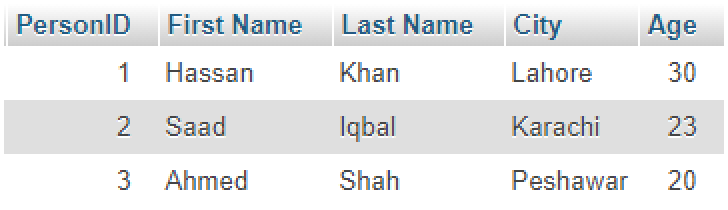
The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

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.

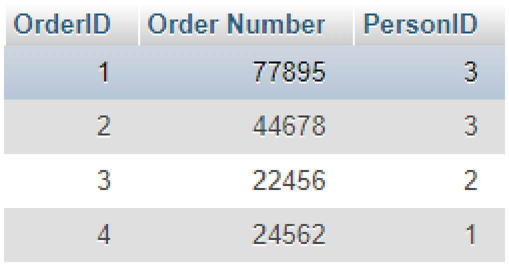
Look at the following two tables:

Persons Table



PersonID	First Name	Last Name	City	Age
1	Hassan	Khan	Lahore	30
2	Saad	Iqbal	Karachi	23
3	Ahmed	Shah	Peshawar	20

Orders Table



OrderID	Order Number	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.

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 (OrderID),
   FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

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

FOREIGN KEY on ALTER TABLE

To create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created, use the following SQL:

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

To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

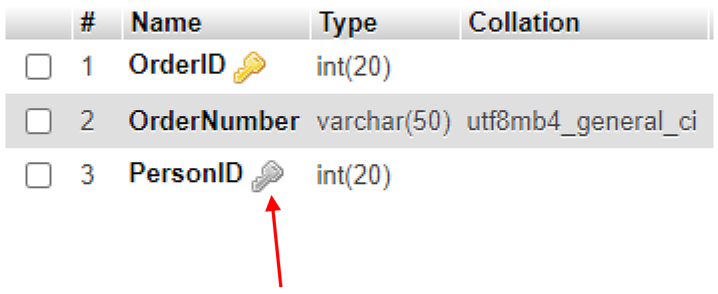
```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

DROP a FOREIGN KEY Constraint

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

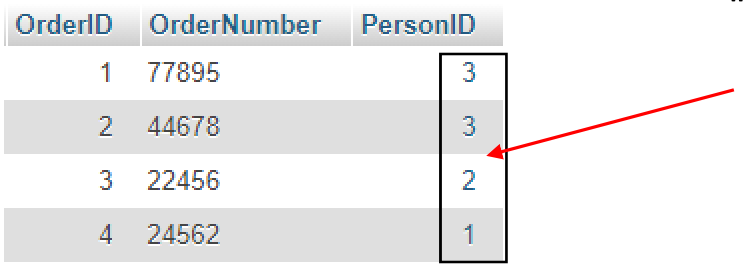
```
ALTER TABLE Orders
DROP FOREIGN KEY FK_PersonOrder;
```

Example:



	#	Name	Type	Collation
☐	1	OrderID	int(20)	
☐	2	OrderNumber	varchar(50)	utf8mb4_general_ci
☐	3	PersonID	int(20)	

Foreign Key Constraint



OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Values color turned into blue which means that the foreign key constraint has been applied successfully. Move the cursor on the value to see the effect of foreign key constraint.

MySQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a column it will allow only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

CHECK on CREATE TABLE

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older:

```
CREATE TABLE Persons (
  PersonID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   CHECK (Age>=18)
);
```

To allow naming of a CHECK constraint, use the following SQL syntax:

```
CREATE TABLE Persons (
  PersonID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   City varchar(255),
   CONSTRAINT CHK_Person CHECK (Age>=18)
);
```

CHECK on ALTER TABLE

To create a CHECK constraint on the "Age" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

To allow naming of a CHECK constraint, use the following SQL syntax:

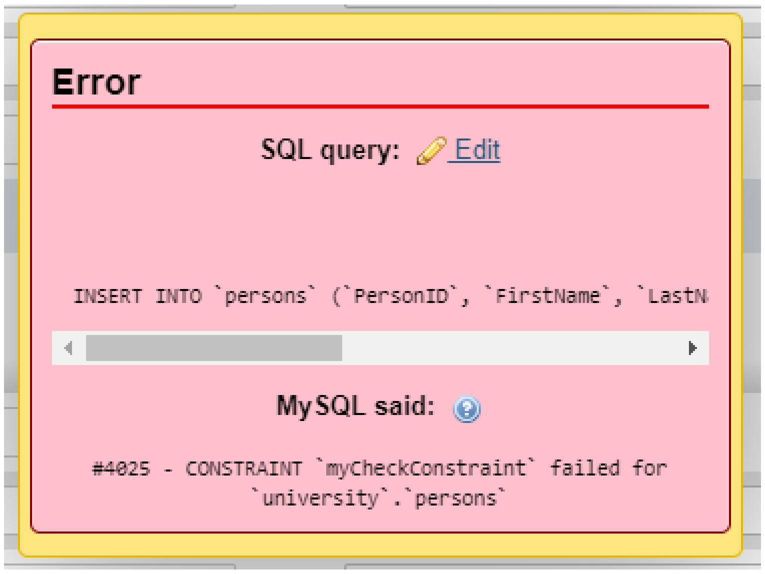
```
ALTER TABLE Persons
ADD CONSTRAINT myCheckConstraint CHECK (Age>=18);
```

DROP a CHECK Constraint

To drop a CHECK constraint, use the following SQL:

```
ALTER TABLE Persons
DROP CONTRAINT myCheckConstraint;
```

Example:



Error

SQL query: Edit

```
INSERT INTO `persons` (`PersonID`, `FirstName`, `LastN
```

MySQL said:

```
#4025 - CONSTRAINT `myCheckConstraint` failed for
`university`.`persons`
```

MySQL DEFAULT Constraint

The DEFAULT constraint is used to set a default value for a column.

The default value will be added to all new records, if no other value is specified.

DEFAULT on CREATE TABLE

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

```
CREATE TABLE Persons (
  PersonID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   City varchar(255) DEFAULT 'Lahore'
);
```

DEFAULT on ALTER TABLE

To create a DEFAULT constraint on the "City" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ALTER City SET DEFAULT 'Lahore';
```

DROP a DEFAULT Constraint

To drop a DEFAULT constraint, use the following SQL:

```
ALTER TABLE Persons
ALTER City DROP DEFAULT;
```

THE END