cd c:\xampp\mysql\bin

mysql –u root

To see the databases

show databases;

To create a database

create database <DB Name>;

*mysql> **CREATE DATABASE new;***

To use a database

use <DB Name>;

```
mysql> USE new
Database changed
```

Creating table

CREATE TABLE *table_name*
(
column_name1 data_type(*size*) *constraint_name*,
column_name2 data_type(*size*) *constraint_name*,
column_name3 data_type(*size*) *constraint_name*,
....
);

In SQL, we have the following constraints:

- **NOT NULL -** Indicates that a column cannot store NULL value
- **UNIQUE -** Ensures that each row for a column must have a unique value

- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly
- **FOREIGN KEY** - Ensure the referential integrity of the data in one table to match values in another table
- **CHECK** - Ensures that the value in a column meets a specific condition
- **DEFAULT** - Specifies a default value for a column

Example

```
CREATE TABLE PersonsNotNull
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

SQL UNIQUE Constraint

The UNIQUE constraint uniquely identifies each record in a database table.

The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it.

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

SQL UNIQUE Constraint on CREATE TABLE

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

```
CREATE TABLE PersonU
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
);
```

SQL UNIQUE Constraint on ALTER TABLE

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

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

```
ALTER TABLE Persons
ADD CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
```

To DROP a UNIQUE Constraint

To drop a UNIQUE constraint, use the following SQL:

```
ALTER TABLE Persons
DROP INDEX uc_PersonID
```

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain UNIQUE values.

A primary key column cannot contain NULL values.

Most tables should have a primary key, and each table can have only ONE primary key.

SQL PRIMARY KEY Constraint on CREATE TABLE

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

CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), PRIMARY KEY (P_Id))	CREATE TABLE Persons (P_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName))

SQL PRIMARY KEY Constraint on ALTER TABLE

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

ALTER TABLE Persons ADD PRIMARY KEY (P_Id)	ALTER TABLE Persons ADD CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)

To DROP a PRIMARY KEY Constraint

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

ALTER TABLE Persons DROP PRIMARY KEY	ALTER TABLE Persons DROP CONSTRAINT pk_PersonID

```
CREATE TABLE Persons
(
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);
```

SQL FOREIGN KEY Constraint

A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

Let's illustrate the foreign key with an example. Look at the following two tables:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Note that the "P_Id" column in the "Orders" table points to the "P_Id" column in the "Persons" table.

The "P_Id" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

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

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

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

SQL FOREIGN KEY Constraint on CREATE TABLE

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

MySQL:

CREATE TABLE Orders (O_Id int NOT NULL, OrderNo int NOT NULL, P_Id int, PRIMARY KEY (O_Id), FOREIGN KEY (P_Id) REFERENCES Persons(P_Id))	CREATE TABLE Orders (O_Id int NOT NULL, OrderNo int NOT NULL, P_Id int, PRIMARY KEY (O_Id), CONSTRAINT fk_PerOrders FOREIGN KEY (P_Id) REFERENCES Persons(P_Id))

SQL FOREIGN KEY Constraint on ALTER TABLE

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

```
ALTER TABLE Orders
ADD FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

```
ALTER TABLE Orders
ADD CONSTRAINT fk_PerOrders
FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

To DROP a FOREIGN KEY Constraint

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

MySQL:

```
ALTER TABLE Orders
DROP FOREIGN KEY fk_PerOrders
```

Others:

Check, Default, Create Index, Drop, Alter, Auto Increment, Views, Dates, Null values, Null Functions, Data Types,

SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

SQL INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form does not specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

The second form specifies both the column names and the values to be inserted:

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

Example

```
INSERT INTO Persons(P_ID, LastName, FirstName, Address, City)
VALUES ('123','Tom','Skagen','Stavanger','Norway');
```

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal','Tom B. Erichsen','Skagen 21','Stavanger','4006','Norway');
```

SQL UPDATE Statement

Example

Change the value of the "City" column of a record in the "Customers" table:

```
UPDATE Customers
SET City='Hamburg'
WHERE CustomerID=1;
```

Some of the Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index