

WEB DEVELOPMENT WITH PHP

COURSE #08



LET'S RECAP

COURSE 7 WAS ABOUT:

- ◆ Cookies
- ◆ Session variables
- ◆ MySQL - SELECT statement
- ◆ PHP + MySQL

TOPICS

COURSE #08

1. MySQL - CREATE TABLE
2. MySQL - INSERT, UPDATE, DELETE
3. MySQL - JOIN
4. MySQL - Altering tables
5. MySQL - Aggregate functions
6. MySQL - Advanced queries
7. Homework

1. MYSQL - CREATE TABLES

LET'S START FROM THE BEGINNING

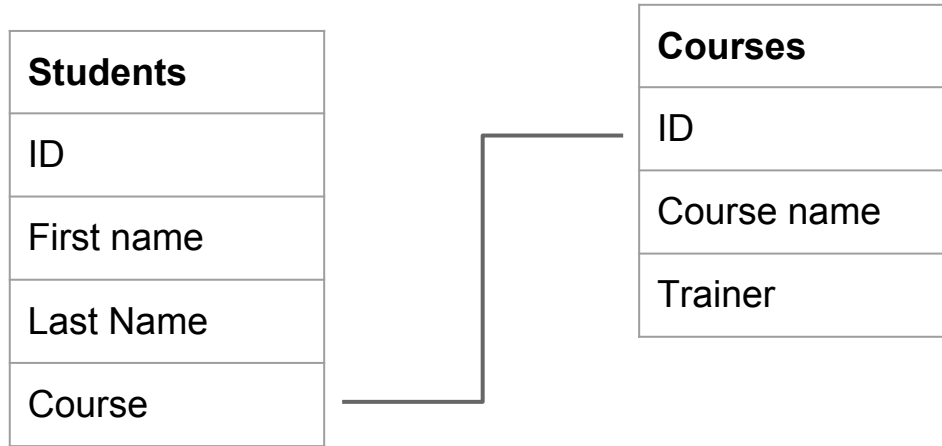
STUDENTS AND COURSES

We'll create a table for Students and another one for Courses and we need to tell MySQL what type of information each of them contains.

But first, we need to "design" it. This is done before actually writing SQL code to create tables, it's a process to identify all the tables you need and the fields for each table.

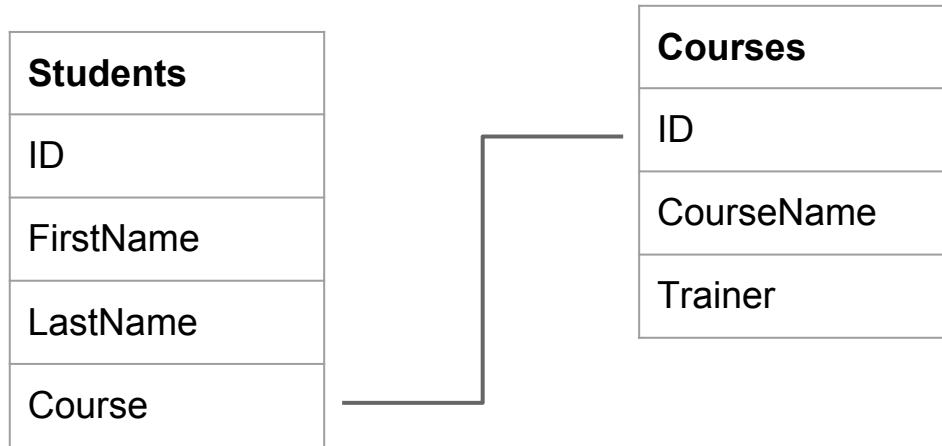
MYSQL DATABASE DESIGN

Let's design the Students and Courses tables:



LET'S MAKE THEM MYSQL FRIENDLY

First of all, the fields name should not contains spaces. You should use only letter, digits and the underscore.



FIELD TYPES

We also need to tell MySQL what type of information we'll put in each field.

INT (TINYINT, SMALLINT, BIGINT)

FLOAT

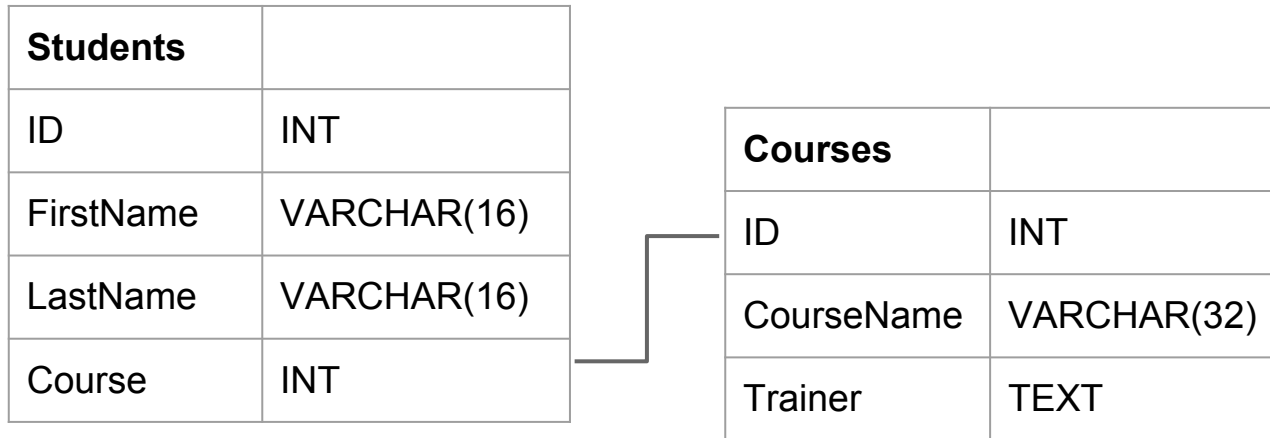
TEXT (VARCHAR, CHAR, LONGTEXT)

DATETIME

BOOLEAN

FIELD TYPES

Let's add the field types



CODING TIME

LET'S FIRST CREATE A NEW DATABASE

Open phpMyAdmin and create a new database.

```
CREATE DATABASE school;
```



CODING TIME

CREATE THE TABLES

Let's start with the Courses table;

```
CREATE TABLE Courses (  
    ID INT NOT NULL AUTO_INCREMENT,  
    CourseName VARCHAR(32),  
    Trainer TEXT,  
    PRIMARY KEY (ID)  
);
```



CODING TIME

KEYWORDS WE USED

AUTO_INCREMENT MySQL will automatically set the value of this field, starting from 1 and increment it for each new row.

NOT NULL This field can't have an empty value, MySQL will trigger an error if you don't provide a value.

PRIMARY KEY MySQL will not allow duplicate values for this field.

CODING TIME

CREATE THE TABLES

Now let's create the Students table:

```
CREATE TABLE Students (  
    ID INT NOT NULL AUTO_INCREMENT,  
    FirstName VARCHAR(16),  
    LastName VARCHAR(16),  
    Course INT,  
    PRIMARY KEY (ID)  
);
```



2. MYSQL - INSERT, UPDATE, DELETE

INSERTING RECORDS

THE INSERT STATEMENT

```
INSERT INTO TableName (  
    Field1, Field2, Field3  
)  
VALUES (  
    Value1, Value2, Value3  
);
```

Your turn, let's insert a course in the Courses table.

INSERTING RECORDS

LET'S ADD A COURSE

```
INSERT INTO Courses (  
    CourseName, Trainer  
)  
VALUES (  
    'Introduction in QA', 'Mike Testovici'  
);
```



INSERTING RECORDS

WE SHOULD NOW HAVE A RECORD IN THE DATABASE

Notice that the ID was filled by MySQL automatically.



The screenshot shows a database management interface. At the top, there is a '+ Options' button and a toolbar with a left arrow, a 'T' icon, and a right arrow. Below this is a table with three columns: 'ID', 'CourseName', and 'Trainer'. The first row of the table contains the values '1', 'Introduction in QA', and 'Mike Testovici'. Below the table, there is a toolbar with a selection icon, a 'Check All' button, and a 'With selected:' label. To the right of the label are three buttons: 'Change' (with a pencil icon), 'Delete' (with a red minus icon), and 'Export' (with a document and arrow icon).

ID	CourseName	Trainer
1	Introduction in QA	Mike Testovici

Now insert another course of your choice.

UPDATING RECORDS

THE UPDATE STATEMENT

```
UPDATE TableName
SET
    Field1 = Value1,
    Field2 = Value2
WHERE
    Field3 = Value3;
```

TIPS & TRICKS



WARNING

When writing UPDATE and DELETE queries, don't forget the ...

WHERE

UPDATING RECORDS

NOW, LET'S UPDATE A COURSE

```
UPDATE Courses
```

```
SET
```

```
    CourseName = 'Introduction in testing',
```

```
    Trainer = 'Michael Testovici'
```

```
WHERE
```

```
    ID = 1;
```



UPDATING RECORDS

WHAT WE SHOULD HAVE NOW:

+ Options

		ID	CourseName	Trainer
<input type="checkbox"/>	 Edit  Copy  Delete	1	Introduction in testing	Michael Testovici
<input type="checkbox"/>	 Edit  Copy  Delete	2	Web development with PHP	Mark Zuckerberg

 ☐ Check All With selected:  Change  Delete  Export

DELETING RECORDS

THE DELETE STATEMENT

```
DELETE FROM TableName  
WHERE Field1 = Value1;
```

TIPS & TRICKS



WARNING

When writing UPDATE and DELETE queries, don't forget the ...

WHERE

DELETING RECORDS

FIRST, LET'S ADD ANOTHER COURSE






```
INSERT INTO Courses (  
    CourseName, Trainer  
)  
VALUES (  
    'Mobile development', 'Steve Jobs'  
);
```





DELETING RECORDS

WE SHOULD HAVE 3 RECORDS:

+ Options

		ID	CourseName	Trainer
<input type="checkbox"/>  Edit  Copy  Delete		1	Introduction in testing	Michael Testovici
<input type="checkbox"/>  Edit  Copy  Delete		2	Web development with PHP	Mark Zuckerberg
<input type="checkbox"/>  Edit  Copy  Delete		3	Mobile development	Steve Jobs

 ☐ Check All With selected:  Change  Delete  Export

DELETING RECORDS

NOW, LET'S DELETE A COURSE





```
DELETE FROM Courses  
WHERE ID = 2;
```





DELETING RECORDS

WE'RE LEFT WITH 2 RECORDS:

+ Options

		ID	CourseName	Trainer
<input type="checkbox"/>	 Edit  Copy  Delete	1	Introduction in testing	Michael Testovici
<input type="checkbox"/>	 Edit  Copy  Delete	3	Mobile development	Steve Jobs

 ☐ Check All *With selected:*  Change  Delete  Export

MYSQL RECAP

WHAT WE SHOULD KNOW SO FAR

- ✓ How to create MySQL tables (the CREATE TABLE statement)
- ✓ How to insert data into MySQL (the INSERT statement)
- ✓ How to update data from MySQL (the UPDATE statement)
- ✓ How to delete data from MySQL (the DELETE statement)
- ✓ How to read data from MySQL (the SELECT statement)
- ✓ ... and how to run all these from PHP

3. MYSQL - JOIN

JOINING TABLES




LET'S PUT SAME DATA IN THE STUDENTS TABLE FIRST



```
INSERT INTO Students (  
    FirstName, LastName, Course  
)  
VALUES  
    ('John', 'Doe', 1),  
    ('James', 'Dean', 1),  
    ('Mary', 'Poppins', 3),  
    ('Forrest', 'Gump', 3);
```

JOINING TABLES

WE SHOULD HAVE 4 STUDENTS:

+ Options

				ID	FirstName	LastName	Course
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	John	Doe	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	James	Dean	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	Mary	Poppins	3
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	Forrest	Gump	3

 ☐ Check All With selected:  Change  Delete  Export

JOINING TABLES

LET'S FETCH SOME STUDENTS

Create an SQL query that reads the first name and last name of all the students.

```
SELECT FirstName, LastName  
FROM Students;
```


JOINING TABLES

LET'S SAY WE ALSO WANT THE COURSE NAME

How we'll do this:

- We select the student's first and last name from the Students table
- For each student we know the Course ID, so we can use it to link to the course in the Courses table
- We select the name of the course from the Courses table

JOINING TABLES

THE JOIN STATEMENT

```
SELECT
    Table1.Field1, Table1.Field2, Table2.Field3
FROM
    Table1
    JOIN Table2 ON Table1.Field4 = Table2.Field5
```

JOINING TABLES

LET'S PUT IN PRACTICE

```
SELECT
    Students.FirstName, Students.LastName,
    Courses.CourseName
FROM
    Students
JOIN Courses ON Students.Course = Courses.ID;
```



JOINING TABLES

LET'S PUT IN PRACTICE

+ Options

FirstName	LastName	CourseName
John	Doe	Introduction in testing
James	Dean	Introduction in testing
Mary	Poppins	Mobile development
Forrest	Gump	Mobile development



JOINING TABLES

WE CAN USE ALL THE OPTIONS FOR THE SELECT STATEMENT

```
SELECT
    Students.FirstName, Students.LastName,
    Courses.CourseName
FROM
    Students
    JOIN Courses ON Students.Course = Courses.ID
WHERE Students.Course = 3
ORDER BY Students.LastName DESC
LIMIT 1
```



JOINING TABLES

TYPES OF JOIN

- INNER JOIN Returns all rows when there is at least one match in BOTH tables.
- LEFT JOIN Return all rows from the left table, and the matched rows from the right table.
- RIGHT JOIN Return all rows from the right table, and the matched rows from the left table.
- FULL JOIN Return all rows when there is a match in ONE of the tables.

4. MYSQL - ALTERING TABLES

ALTERING TABLES

WHAT OPERATIONS CAN WE DO ON A TABLE

- Add a new field
- Update an existing field
- Delete a field
- Delete the entire table
- etc.



ALTER TABLE

ALTERING TABLES

ADDING A NEW FIELD

Let's add a new field, Score, to the Students table.

```
ALTER TABLE Students  
ADD Score INT;
```

ALTERING TABLES

ADDING A NEW FIELD

+ Options

		ID	FirstName	LastName	Course	Score
<input type="checkbox"/>	Edit Copy Delete	1	John	Doe	1	NULL
<input type="checkbox"/>	Edit Copy Delete	2	James	Dean	1	NULL
<input type="checkbox"/>	Edit Copy Delete	3	Mary	Poppins	3	NULL
<input type="checkbox"/>	Edit Copy Delete	4	Forrest	Gump	3	NULL

☐ Check All With selected: Change Delete Export

ALTERING TABLES

ADDING A NEW FIELD

How do we add the scores for students?

```
UPDATE Students  
SET Score = 7  
WHERE ID = 1;
```

ALTERING TABLES

HOME STUDY

Study and practice at home:

- Updating an existing field
- Deleting an existing field
- Deleting a table

<http://dev.mysql.com/doc/refman/5.7/en/alter-table.html>

5. MYSQL - AGGREGATE FUNCTIONS

AGGREGATE FUNCTIONS

WHAT ARE AGGREGATE FUNCTIONS

An aggregate function takes many values and returns a single value, computed from the initial values:

- COUNT
- SUM
- MAX, MIN, AVG
- DISTINCT

AGGREGATE FUNCTIONS

COUNT

Returns the count of rows which match the conditions.

Let's say we want to know how many students have joined the QA course.

```
SELECT COUNT(*) AS QAStudentsCount  
FROM Students  
WHERE Course = 1;
```

AGGREGATE FUNCTIONS

GROUPING RESULTS

By default, the aggregate functions are applied on ALL results.

```
SELECT COUNT(*) AS QASStudentsCount  
FROM Students;
```

+ Options

QASStudentsCount

4

AGGREGATE FUNCTIONS

GROUPING RESULTS

+ Options

		ID	FirstName	LastName	Course	Score
<input type="checkbox"/>	Edit Copy Delete	1	John	Doe	1	6
<input type="checkbox"/>	Edit Copy Delete	2	James	Dean	1	3
<input type="checkbox"/>	Edit Copy Delete	3	Mary	Poppins	3	9
<input type="checkbox"/>	Edit Copy Delete	4	Forrest	Gump	3	7

☐ Check All With selected: Change Delete Export

If we want to display the aggregate value for each course we would need to tell MySQL to GROUP the results by the Course field.

AGGREGATE FUNCTIONS

GROUPING RESULTS

```
SELECT Course, COUNT(*) AS QAStudentsCount  
FROM Students  
GROUP BY Course;
```

+ Options

Course	QAStudentsCount
1	2
3	2

AGGREGATE FUNCTIONS

SUM, AVG, MAX, MIN

Let's say we want to calculate the average score across all students.

```
SELECT AVG(Score) AS AverageScore  
FROM Students;
```

Similarly you can calculate the minimum score, the maximum score or the sum of scores.

6. MYSQL - ADVANCED QUERIES

ADVANCED QUERIES

COMBINING STUFF

We can combine SELECT, JOIN and aggregate functions to create more complex queries.

For example, we might want to display a table which has:

- Course names, ordered alphabetically
- And the average score of the students enrolled in each course

ADVANCED QUERIES

IT SHOULD DISPLAY SOMETHING LIKE THIS:

+ Options

CourseName ▲ 1	AverageScore
Introduction in testing	4.5000
Mobile development	8.0000

AGGREGATE FUNCTIONS

LET'S BUILD IT STEP BY STEP

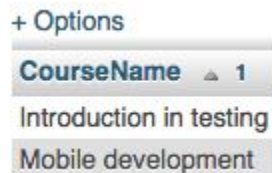
First, let's display the course names, alphabetically.

```
SELECT CourseName  
FROM  
    Courses  
ORDER BY CourseName;
```

AGGREGATE FUNCTIONS

LET'S BUILD IT STEP BY STEP

First, let's display the course names, alphabetically.



+ Options	
CourseName	▲ 1
Introduction in testing	
Mobile development	

AGGREGATE FUNCTIONS

LET'S BUILD IT STEP BY STEP

Now, let's use a RIGHT JOIN to also select all the students and display the score for each students.

```
SELECT CourseName, Score
FROM
    Courses
    JOIN Students ON Courses.ID = Students.Course
ORDER BY CourseName;
```

AGGREGATE FUNCTIONS

LET'S BUILD IT STEP BY STEP

Now, let's use a JOIN to also select all the students and display the score for each students.

+ Options

CourseName ▲ 1	Score
Introduction in testing	6
Introduction in testing	3
Mobile development	9
Mobile development	7

AGGREGATE FUNCTIONS

LET'S BUILD IT STEP BY STEP

And finally, group the results by Course and calculate the average score.

```
SELECT CourseName, AVG(Score) as AverageScore
FROM
    Courses
    JOIN Students ON Courses.ID = Students.Course
GROUP BY Course
ORDER BY CourseName
```

AGGREGATE FUNCTIONS

LET'S BUILD IT STEP BY STEP

And finally, group the results by Course and calculate the average score.

+ Options

CourseName ▲ 1	AverageScore
Introduction in testing	4.5000
Mobile development	8.0000

7. HOMEWORK

HOMEWORK

COURSES MANAGEMENT PAGE

Create a small site where you can manage the data in the courses table.

The site should:

- list all the courses in the database
- allow you to add a new course
- allow you to delete a course
- allow you to edit a course (*bonus point*)

HOMEWORK

1. LIST COURSES

- Create a page called list.php
- Read all the courses from the database using a SQL query
- Loop through all the courses and put them in an array
- Loop through the array and create a HTML table
- Below the table add a link to a page called add.php

(see next page for a wireframe)

HOMEWORK

1. LIST COURSES

Course name	Trainer	Operations	
Introduction in testing	Michael Testovici	Edit	Delete
Mobile development	Steve Jobs	Edit	Delete

[Add a new course](#)

HOMEWORK

2. ADD COURSE

- Create a page called add.php
- The page should contain a form with two text fields
 - Course name
 - Trainer
- On submit, verify that both fields are filled
- Create a SQL query to insert the new course in the database
- Redirect to list.php

(see next page for a wireframe)

HOMEWORK

2. ADD COURSE

Course name

Trainer

[Add course](#)

HOMEWORK

3. DELETE COURSE

- In list.php, link the delete links to delete.php?ID=[course ID]
- Create a page called delete.php
- The page should accept a \$_GET parameter called ID
- If the parameter is present, create a SQL query to delete the course having this parameter
- Redirect to list.php

HOMEWORK

4. EDIT COURSE (BONUS)

- In list.php, link the edit links to edit.php?ID=[course ID]
- Create a page called edit.php
- The page should accept a \$_GET parameter called ID
- If the parameter is present, create a SQL query to read the course details from database

HOMEWORK

4. EDIT COURSE (BONUS) - CONTINUED

- Create a form with:
 - Two text fields, pre-filled with the values from the database
 - Course name
 - Trainer
 - A hidden field pre-filled with the course ID
- On submit, verify if the text fields are filled (not empty)
- Then create a SQL query which updates the record which has the ID from the hidden field with the values received by form
- Redirect to list.php

HOMEWORK

5. DISPLAY STUDENTS COUNT (**BONUS**)

In the list.php page, add a new column which displays the number of students for each course.

Hint: Use a JOIN and a COUNT in the original SELECT query which retrieves all the courses.