

Relational Databases

Integration of PHP and MySQL

Web Development and Security (ZEIT3119)

Week 7

Dr. Reza Rafeh

Revision

What are the common uses of PHP?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- It is embedded in HTML files but all the PHP tags are replaced by the server before anything is sent to the web browser

Revision

What is the difference between static and dynamic pages?

Static Page

- In static websites, content can't be changed after running the script
- You cannot change anything in the site as it is predefined
- Example: Wikipedia

Dynamic Page

- Content of script can be changed at the run time
- Its content is regenerated every time a user visits or reloads.
- Example: E-bay, Amazon, Netflix, shopping website.

Revision

PHP forms

GET vs POST

- \$_GET is an array of variable names and values sent by the HTTP GET method via the URL parameters.
 - Form data is visible to everyone
 - It is possible to bookmark the page
 - The limitation is about 2000 characters
 - May be used for sending non-sensitive data (no passwords)
- \$_POST is an array of variables passed to the current script via the HTTP POST method
 - Information is invisible to everyone
 - No limits
 - Supports advanced functionality such as multi-part binary input while uploading files to server

Outline

- Data Modelling (Slides 6-11)
- Relational Databases and SQL (Slides 12-39)
- MySQL (Slides 40-59)
- Integrating PHP and MySQL (Slides 60-71)

Database Design

Before creating and using a database, we need to design it.

We need to consider

- What tables, keys, and constraints are needed?
- What is the database going to be used for?

Conceptual design

- Build a model independent of the choice of DBMS

Logical design

- Create the database in a given DBMS

Physical design

- How the database is stored on hardware

Entity/Relationship Modelling

E/R Modelling is used for conceptual design

- Entities - objects or items of interest
- Attributes - facts about, or properties of, an entity
- Relationships - links between entities

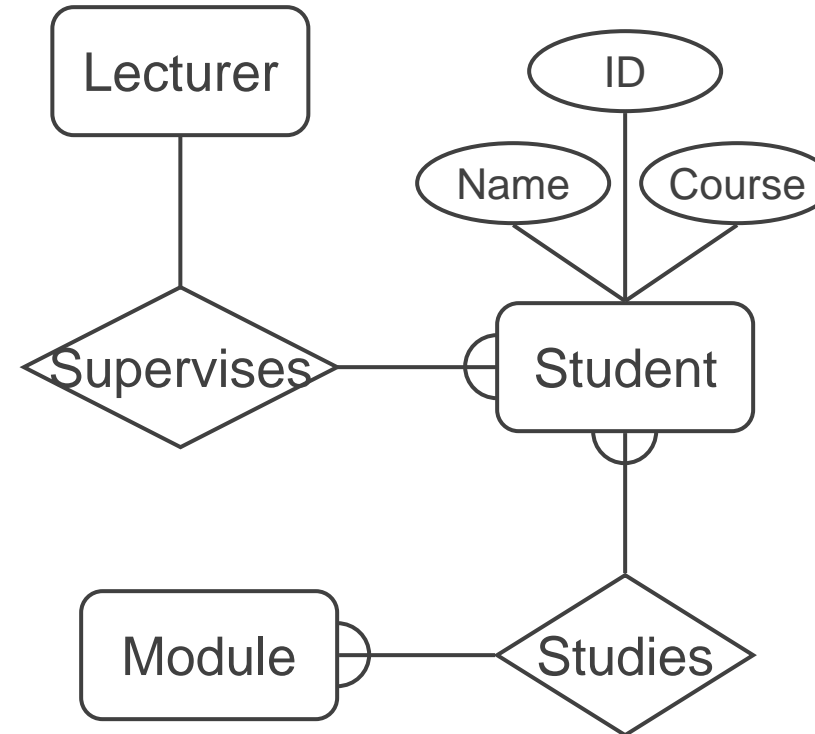
Example

In a University database we may have entities for Students, Modules and Lecturers. Students may have attributes such as their ID, Name, and Course, and can have relationships with Modules (enrolment) and Lecturers (tutor/tutee)

Entity/Relationship Diagrams

E/R Models are often represented as E/R diagrams that

- Give a conceptual view of the database
- Are independent of the choice of DBMS
- Can identify some problems in a design



Entities

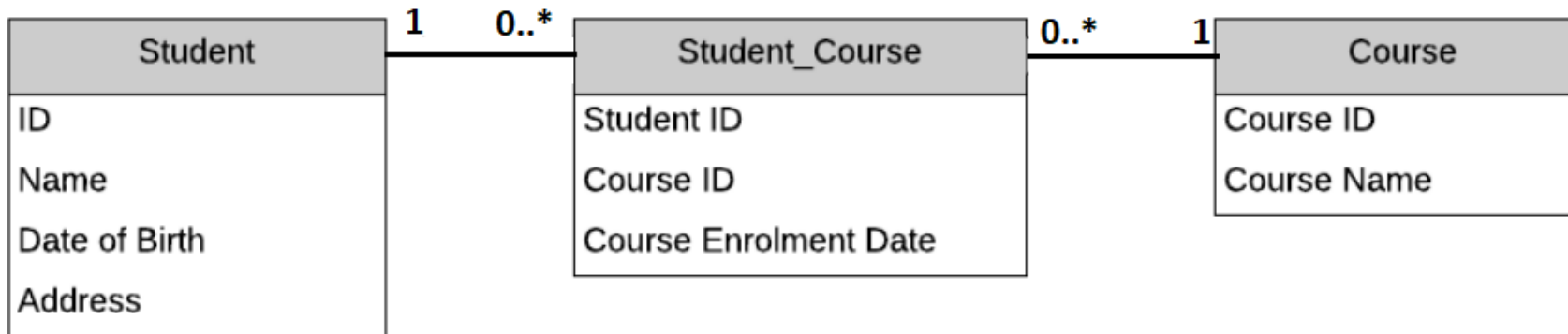
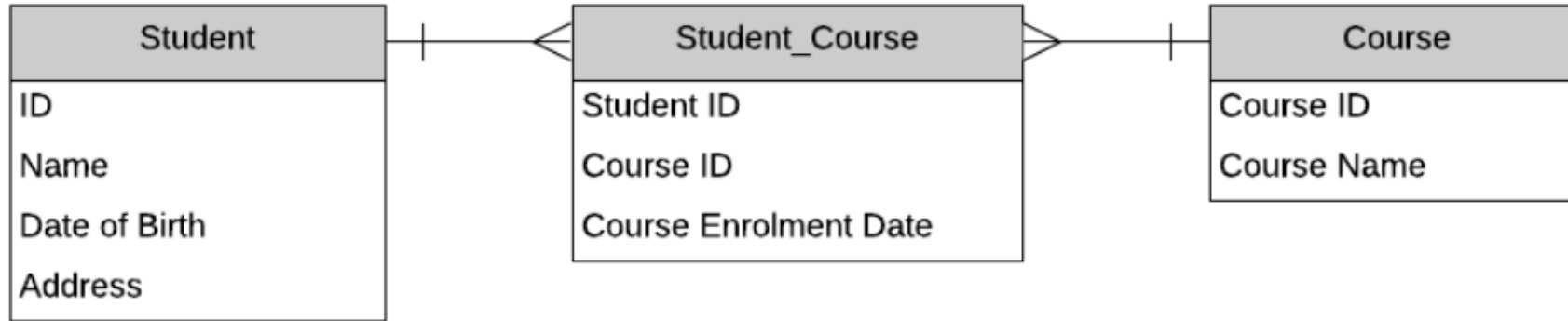
Entities represent objects or things of interest

- Physical things like students, lecturers, employees, products
- More abstract things like modules, orders, courses, projects

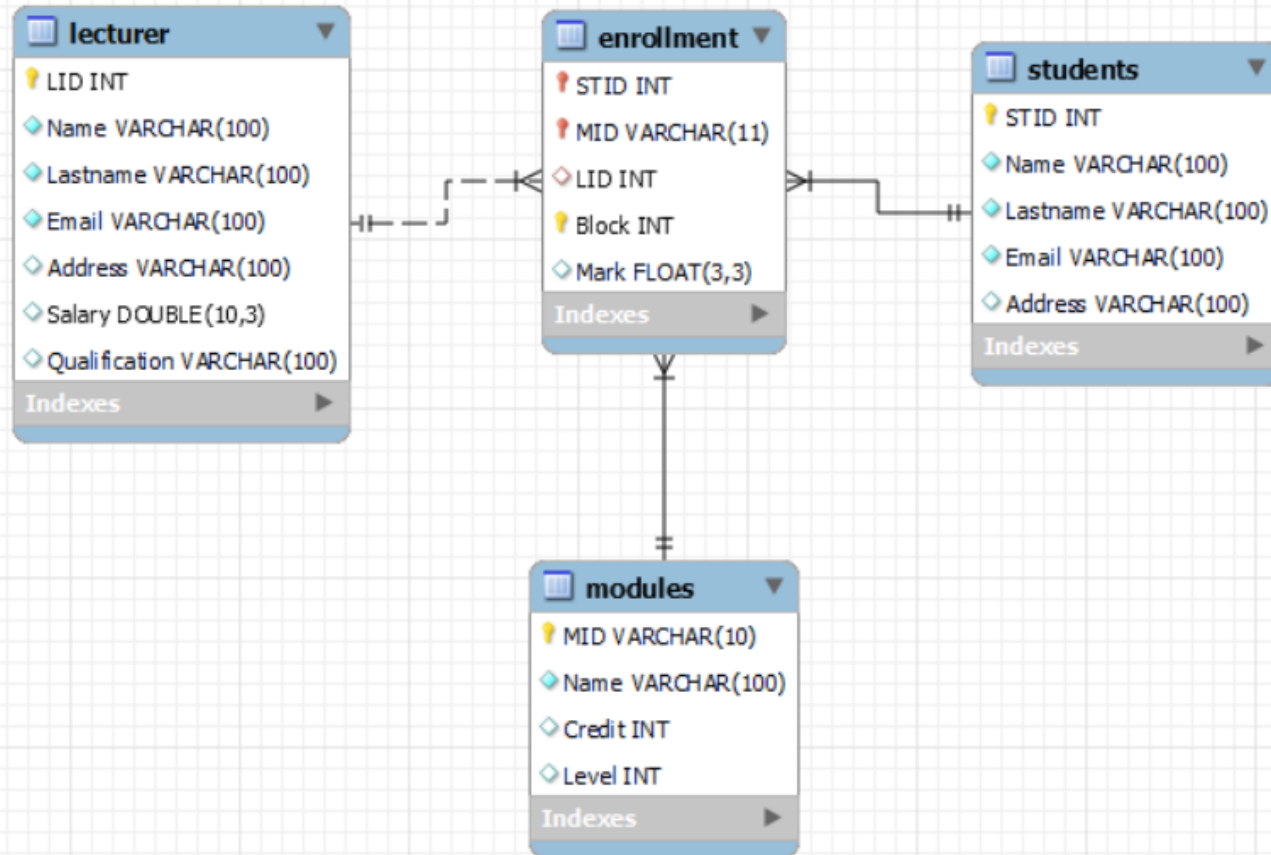
Entities have

- A general type or class, such as Lecturer or Module
- Instances of that particular type, such as Reza Rafeh, Anne Smith are instances of Lecturer
- Attributes (such as name, email address)

ERD – Different Notations



Logical Design



SQL (Structured Query Language)

SQL is not a programming language, but rather a data sublanguage.

SQL is comprised of

- A data definition language (DDL)
 - Used to define database structures
- A data manipulation language (DML)
 - Data definition and updating
 - Data retrieval (Queries)

SQL for Data Definition

The SQL data definition statements include:

- CREATE
 - To create database objects
- ALTER
 - To modify the structure and/or characteristics of database objects
- DROP
 - To delete database objects
- TRUNCATE
 - To delete table data while keeping structure

SQL for Data Definition: CREATE

Creating database

```
CREATE DATABASE EMPLOYMENT;
```

Creating database tables

```
CREATE TABLE EMPLOYEE (  
    EmpID          Integer          PRIMARY KEY,  
    EmpName        Char (25)        NOT NULL  
);
```

Run SQL in MySQL

The screenshot shows the Laragon MySQL interface. The top menu bar includes File, Edit, Search, Query, Tools, Go to, and Help. Below the menu is a toolbar with various icons. The status bar at the top indicates the Host is 127.0.0.1, the Database is enrollment, and the selected Table is employee. The left sidebar shows the database structure, with the 'employee' table selected under the 'enrollment' database. The main editor displays the following SQL query:

```
1 CREATE TABLE EMPLOYEE(  
2   EmpID      Integer      PRIMARY KEY,  
3   EmpName    Char(25)     NOT NULL  
4 );  
5
```

Red annotations highlight the following elements:

- 1. The 'Query #2*' tab in the top right.
- 2. The SQL query text in the main editor.
- 3. The 'Run' button (play icon) in the top toolbar.
- 4. The 'employee' table selected in the left sidebar.
- 5. The 'Run' button in the top toolbar.

SQL for Data Definition: CREATE with CONSTRAINT

Creating database tables with PRIMARY KEY constraints

- The SQL CREATE TABLE statement
- The SQL CONSTRAINT keyword

```
CREATE TABLE EMPLOYEE (  
  
    EmpID            Integer      NOT NULL,  
  
    EmpName          Char (25)    NOT NULL  
  
    CONSTRAINT Emp_PK      PRIMARY KEY (EmpID)  
  
);
```


SQL for Data Definition:

Creating database tables with composite primary keys using PRIMARY KEY constraints

- The SQL CREATE TABLE statement
- The SQL CONSTRAINT keyword

```
CREATE TABLE EMP_SKILL(  
    EmpID          Integer      NOT NULL,  
    SkillID        Integer      NOT NULL,  
    SkillLevel     Integer      NULL,  
    CONSTRAINT EmpSkill_PK PRIMARY KEY  
                        (EmpID, SkillID)  
);
```

SQL for Data Definition:

Creating database tables using PRIMARY KEY and FOREIGN KEY constraints

- The SQL CREATE TABLE statement
- The SQL CONSTRAINT keyword

```
CREATE TABLE EMP_SKILL(  
    EmpID      Integer      NOT NULL,  
    SkillID    Integer      NOT NULL,  
    SkillLevel Integer      NULL,  
    CONSTRAINT EmpSkill_PK PRIMARY KEY (EmpID, SkillID),  
    CONSTRAINT Emp_FK      FOREIGN KEY(EmpID) REFERENCES EMPLOYEE(EmpID),  
    CONSTRAINT Skill_FK    FOREIGN KEY(SkillID) REFERENCES SKILL(SkillID)  
);
```

Adding Data: **INSERT**

To add a row to an existing table, use the INSERT statement.

Non-numeric data must be enclosed in straight (') single quotes.

```
INSERT INTO EMPLOYEE VALUES (91, 'Smither', 12);
```

```
INSERT INTO EMPLOYEE (EmpID, SalaryCode)  
VALUES (62, 11);
```

SQL for Data Retrieval: Displaying All Columns

To show all of the column values for the rows that match the specified criteria, use an asterisk (*).

```
SELECT    *  
  
FROM      EMPLOYEE ;
```

SQL for Data Retrieval: Showing Each Row Only Once

The DISTINCT keyword may be added to the SELECT statement to inhibit duplicate rows from displaying.

```
SELECT  DISTINCT DeptID  
  
FROM    EMPLOYEE;
```

SQL for Data Retrieval:

Specifying Search Criteria

The WHERE clause stipulates the matching criteria for the record that is to be displayed.

```
SELECT      EmpName  
  
FROM        EMPLOYEE  
  
WHERE       DeptID = 15;
```

SQL for Data Retrieval:

A List of Values

The WHERE clause may include the IN keyword to specify that a particular column value must be included in a list of values.

```
SELECT      EmpName
FROM        EMPLOYEE
WHERE       DeptID IN (4, 8, 9);
```

SQL for Data Retrieval:

The Logical NOT Operator

Any criteria statement may be preceded by a NOT operator, which is to say that all information will be shown except that information matching the specified criteria

```
SELECT    EmpName
FROM      EMPLOYEE
WHERE     DeptID NOT IN (4, 8, 9);
```


SQL for Data Retrieval:

Finding Data in a Range of Values

SQL provides a BETWEEN keyword that allows a user to specify a minimum and maximum value on one line.

```
SELECT  EmpName
FROM    EMPLOYEE
WHERE   SalaryCode BETWEEN 10 AND 45;
```

SQL for Data Retrieval:

Allowing for Wildcard Searches

The SQL LIKE keyword allows searches on partial data values.

LIKE can be paired with wildcards to find rows matching a string value.

- Multiple character wildcard character is a percent sign (%).
- Single character wildcard character is an underscore (_).

SQL for Data Retrieval: Wildcard Search Examples

```
SELECT          EmpID
FROM            EMPLOYEE
WHERE           EmpName LIKE 'Wilk%';
```

```
SELECT          EmpID
FROM            EMPLOYEE
WHERE           Phone LIKE '07-____-____';
```

SQL for Data Retrieval: Sorting the Results

Query results may be sorted using the ORDER BY clause.

```
SELECT      *  
  
FROM        EMPLOYEE  
  
ORDER BY    EmpName ;
```

SQL for Data Retrieval:

Built-in SQL Functions

SQL provides several built-in functions:

- COUNT : Counts the number of rows that match the specified criteria
- MIN: Finds the minimum value for a specific column for those rows matching the criteria
- MAX: Finds the maximum value for a specific column for those rows matching the criteria
- SUM: Calculates the sum for a specific column for those rows matching the criteria
- AVG: Calculates the numerical average of a specific column for those rows matching the criteria

SQL for Data Retrieval:

Built-in Function Examples

```
SELECT COUNT (DeptID)
FROM      EMPLOYEE ;
```

```
SELECT MIN (Hours) AS MinimumHours,
        MAX (Hours) AS MaximumHours,
        AVG (Hours) AS AverageHours
FROM      PROJECT
WHERE     ProjID > 7;
```

SQL for Data Retrieval:

Providing Subtotals: GROUP BY

Subtotals may be calculated by using the GROUP BY clause.

The HAVING clause may be used to restrict which data is displayed.

```
SELECT COUNT(CustomerID) , Country  
  
FROM Customers  
  
GROUP BY Country;
```

COUNT(CustomerID)	Country
3	Argentina
2	Austria
2	Belgium

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium
76	Suprêmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium

SQL for Data Retrieval:

Retrieving Information from Multiple Tables

Subqueries

- As stated earlier, the result of a query is a relation. As a result, a query may feed another query. This is called a *subquery*.

Joins

- Another way of combining data is by using a *join* .
 - Join [also called an Inner Join]
 - Left Outer Join
 - Right Outer Join

SQL for Data Retrieval: Subquery Example

```
SELECT EmpName
FROM      EMPLOYEE
WHERE     DeptID in
          (SELECT  DeptID
           FROM     DEPARTMENT
           WHERE    DeptName LIKE 'Account%') ;
```

SQL for Data Retrieval: Join Example

```
SELECT  EmpName
FROM    EMPLOYEE AS E, DEPARTMENT AS D
WHERE   E.DeptID = D.DeptID
        AND D.DeptName LIKE 'Account%';
```

Modifying Data using SQL

Insert

- Will add a new row in a table (already discussed above)

Update

- Will update the data in a table that matches the specified criteria

Delete

- Will delete the data in a table that matches the specified criteria

Modifying Data using SQL:

Changing Data Values: UPDATE

To change the data values in an existing row (or set of rows) use the Update statement.

```
UPDATE      EMPLOYEE  
  
SET         Phone '791-555-1234'  
  
WHERE       EmpID = 29;
```

```
UPDATE      EMPLOYEE  
  
SET         DeptID = 44  
  
WHERE       EmpName LIKE 'Kr%';
```

Modifying Data using SQL:

Deleting Data: DELETE

To delete a row or set of rows from a table use the DELETE statement.

```
DELETE FROM EMPLOYEE
```

```
WHERE EmpID = 29;
```

```
DELETE FROM EMPLOYEE
```

```
WHERE EmpName LIKE 'Kr%';
```

Modifying Data using SQL:

Deleting Database Objects: DROP

To remove unwanted database objects from the database, use the SQL DROP statement.

Warning... The DROP statement will permanently remove the object and all data.

```
DROP TABLE EMPLOYEE;
```

Modifying Data using SQL:

Removing a Constraint: ALTER & DROP

To change the constraints on existing tables, you may need to remove the existing constraints before adding new constraints.

```
ALTER TABLE EMPLOYEE DROP CONSTRAINT EmpFK;
```

MySQL Tutorial

- MySQL is an open-source relational database management system (RDBMS).
- MySQL is based on SQL (Structured Query Language) which is a standard language for managing relational databases.
- It is one of the most popular database systems used by web applications.
- MySQL can be installed on various operating systems such as Windows, Linux, macOS, etc.
- MySQL offers high performance, scalability, and reliability.
- It provides a wide range of features including support for transactions, indexing, triggers, stored procedures, views, and more.

Start Laragon and Database

Type your password

Laragon Full 5.0.0 210523 php-7.4.19-Win32-vc15-x64 [TS] 192.168... Session manager

Menu

© Leo K

The journey of a thousand miles begins with one step.

Start All Web Database

Session name ^ Host

Unnamed	127.0...
---------	----------

Filter ...

Settings Advanced Statistics

Network type: MariaDB or MySQL (TCP/I

Library: libmariadb.dll

Hostname / IP: 127.0.0.1

☐ Prompt for credentials

☐ Use Windows authentication

User: root

Password: [REDACTED]

Port: 3306

☐ Compressed client/server p

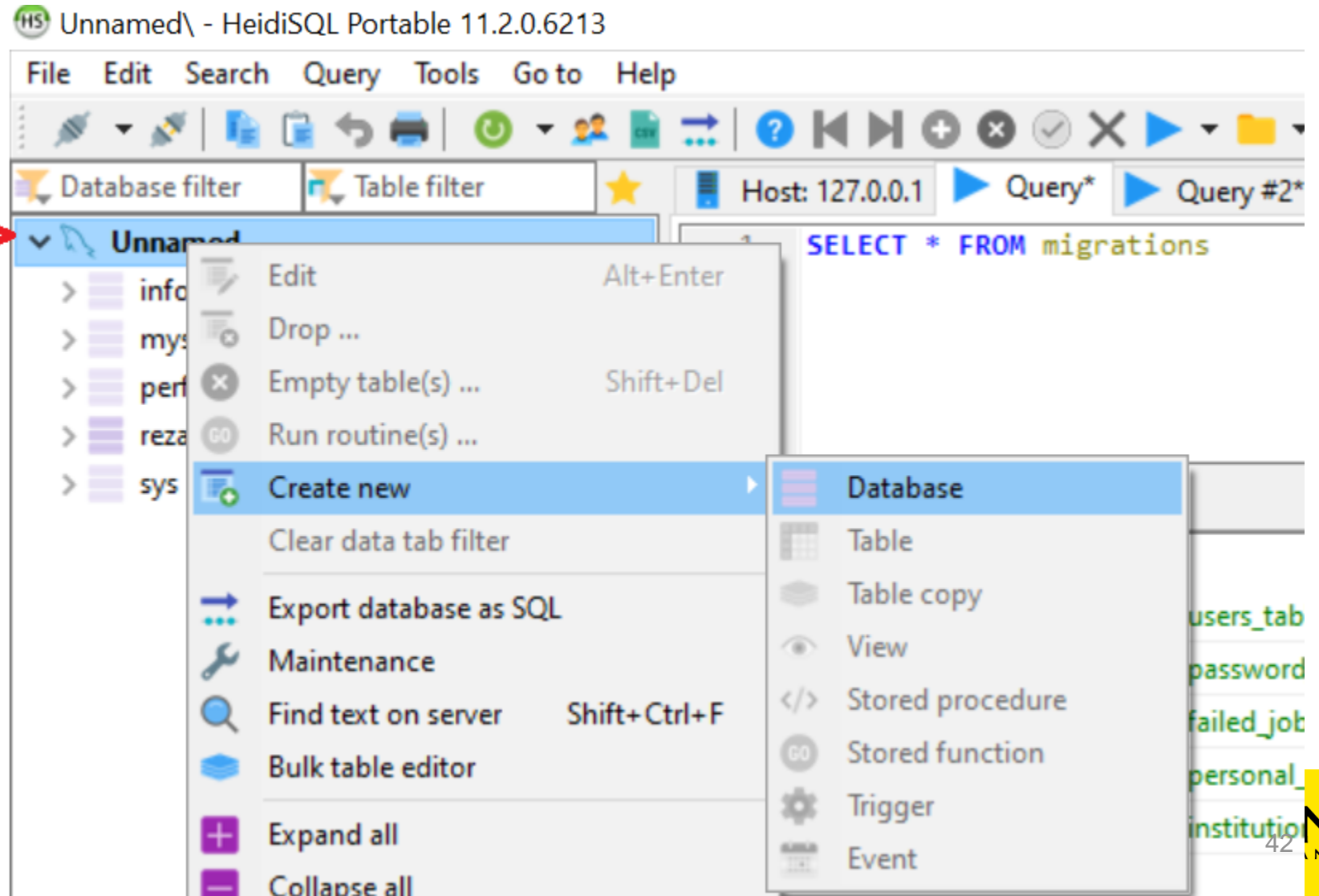
Databases: Separated by semicolon

Comment:

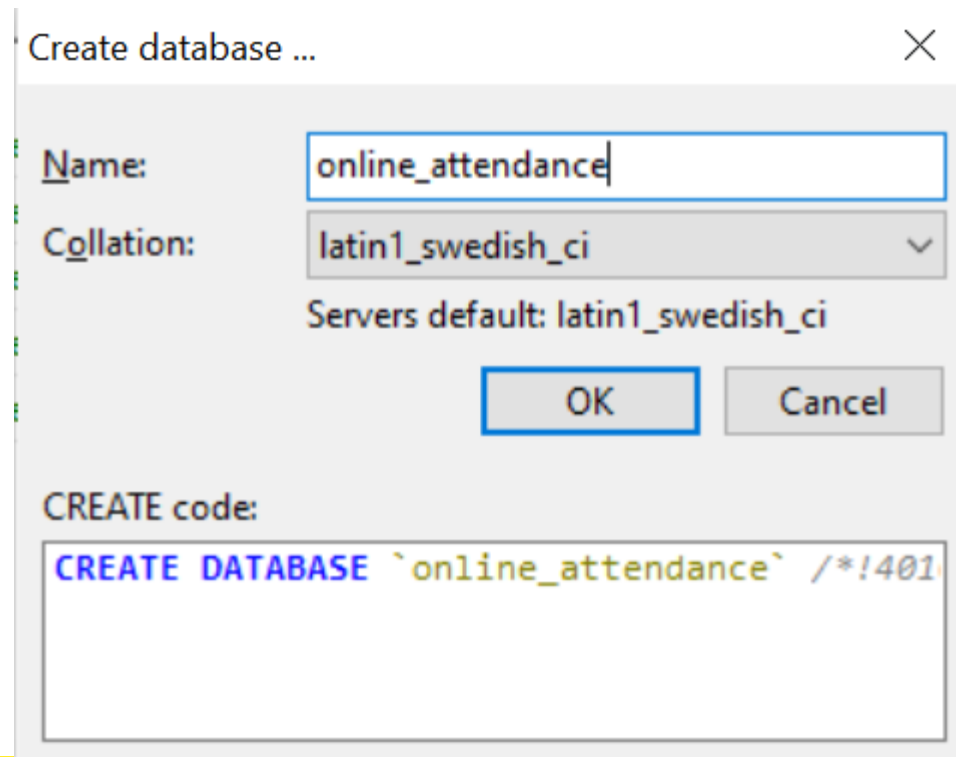
+ New Save Delete Open Cancel

Create a Database

Right Click



Choose a name for the database



Create database ...

Name:

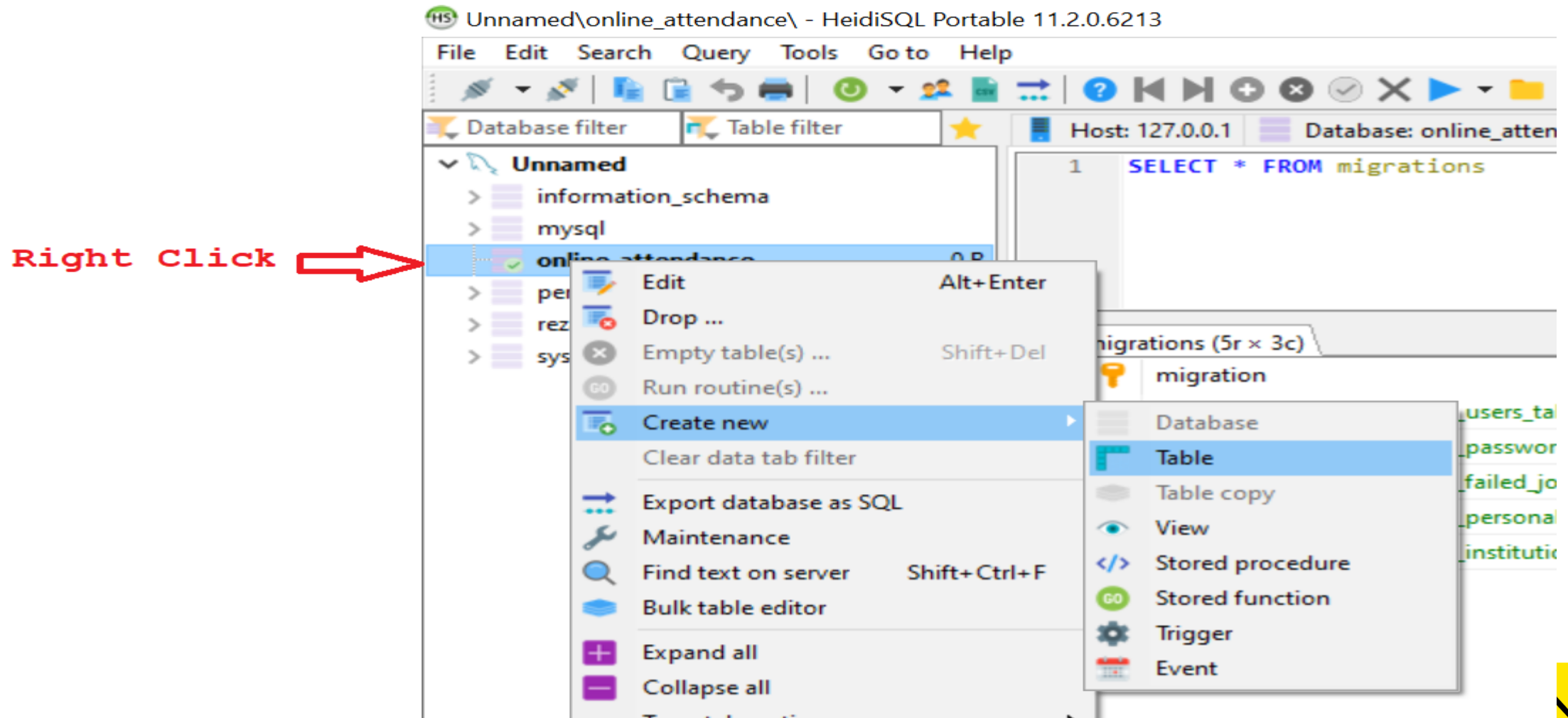
Collation:

Servers default: latin1_swedish_ci

CREATE code:

```
CREATE DATABASE `online_attendance` /*!401
```

Create a Table



Add Columns to the Table

Basic Options Indexes (0) Foreign keys (0) Check constraints (0) Partitions CREATE code

Name: modules

Comment:

Add a new column

Columns: + Add x Remove ▲ Up ▼ Down

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Comment
1	MID	VARCHAR	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
2	Name	VARCHAR	20	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
3	Credit	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	

Choose data type

max length of values

Create a Primary Key

Host: 127.0.0.1 Database: online_attendance

Table filter

n_schema

endance 0 B

ce_schema

Basic Options Indexes (0) Foreign key

Name: modules

Comment:

Columns: + Add - Remove ▲ Up ▼ Down

#	Name	Datatype	Length
1	MID	VARCHAR	10
			20

Right Click →

- Copy Ctrl+C
- Copy selected columns
- Paste columns
- + Add column Ctrl+Ins
- Remove column Ctrl+Del
- ▲ Move up Ctrl+U
- ▼ Move down Ctrl+D
- ⚡ Create new index
- ⚡ Add to index

PRIMARY KEY UNIQUE FULLTEXT SPATIAL

Filter: Regular expression

Help Discard Save

Abstract Error Message CharCode:0 Msg:514 */

Primary Key

Host: 127.0.0.1 Database: online_attendance Table: [Untitled] Query* Query #2* X

Basic Options **Indexes (1)** Foreign keys (0) Check constraints (0) Partitions CREATE code

+ Add
✖ Remove
✖ Clear
▲ Up
▼ Down

Name
▼ PRIMARY KEY
 ◇ MID

Not null No default value

Columns: + Add ✖ Remove ▲ Up ▼ Down

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Co
1	MID	VARCHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default	
2	Name	VARCHAR	20	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
3	Credit	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	

Add Rows to the Table

HS Unnamed\online_attendance\modules\ - HeidiSQL Portable 11.2.0.6213

File Edit Search Query Tools Go to Help

Database filter Table filter Host: 127.0.0.1 Database: online_attendance Table: modules Data

online_attendance.modi >> Next Show all Sorting Colu

MID	Name	Credit
IX607001	Intro App Development	15

Insert new row

Choose Data

Type values

Run SQL Queries

The screenshot shows the HeidiSQL Portable 11.2.0.6213 interface. The title bar indicates the connection is to 'Unnamed\online_attendance\modules'. The menu bar includes File, Edit, Search, Query, Tools, Go to, and Help. The toolbar contains various icons for database operations, with a red arrow pointing to the 'Run query' button (a blue play icon). Below the toolbar, the 'Database filter' and 'Table filter' are visible. The left sidebar shows a tree view of databases: 'Unnamed' (expanded), 'information_schema', 'mysql', 'online_attendance' (selected), and 'modules' (selected). The main query editor contains the text '1 SELECT * FROM modules' and a red arrow pointing to it with the label 'Type query here'. To the right of the query editor is a 'Filter' dropdown menu with options: 'Columns in modules', 'SQL functions', 'SQL keywords', and 'Snippets'. A red arrow points to the 'SQL keywords' option with the label 'Select'. Below the query editor, the 'Query result' is displayed as a table with columns 'MID', 'Name', and 'Credit'. The table contains one row: 'IX607001', 'Intro App Development', and '15'.

Run query

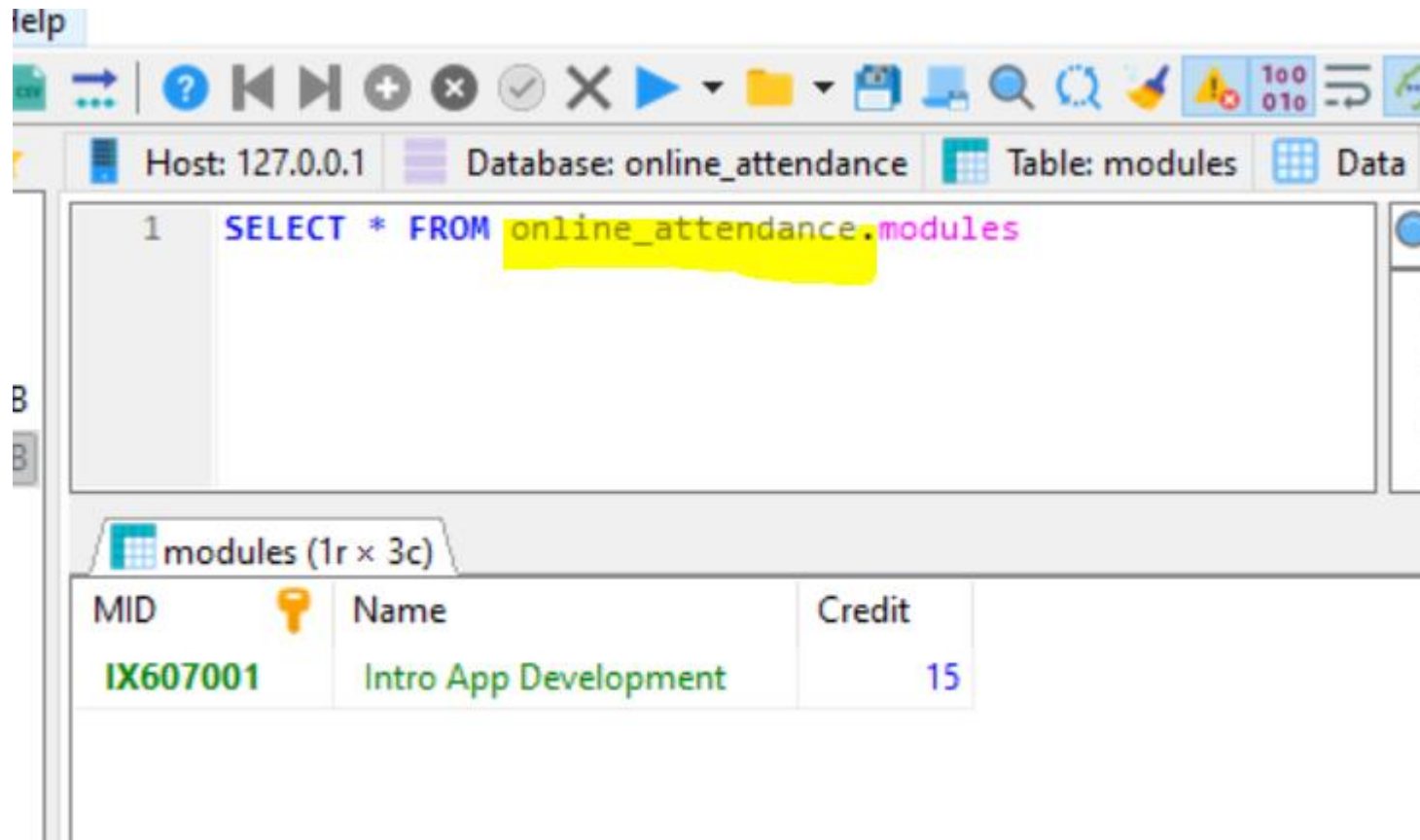
Type query here

Select

Query result

MID	Name	Credit
IX607001	Intro App Development	15

Referring to the Database Name in Queries



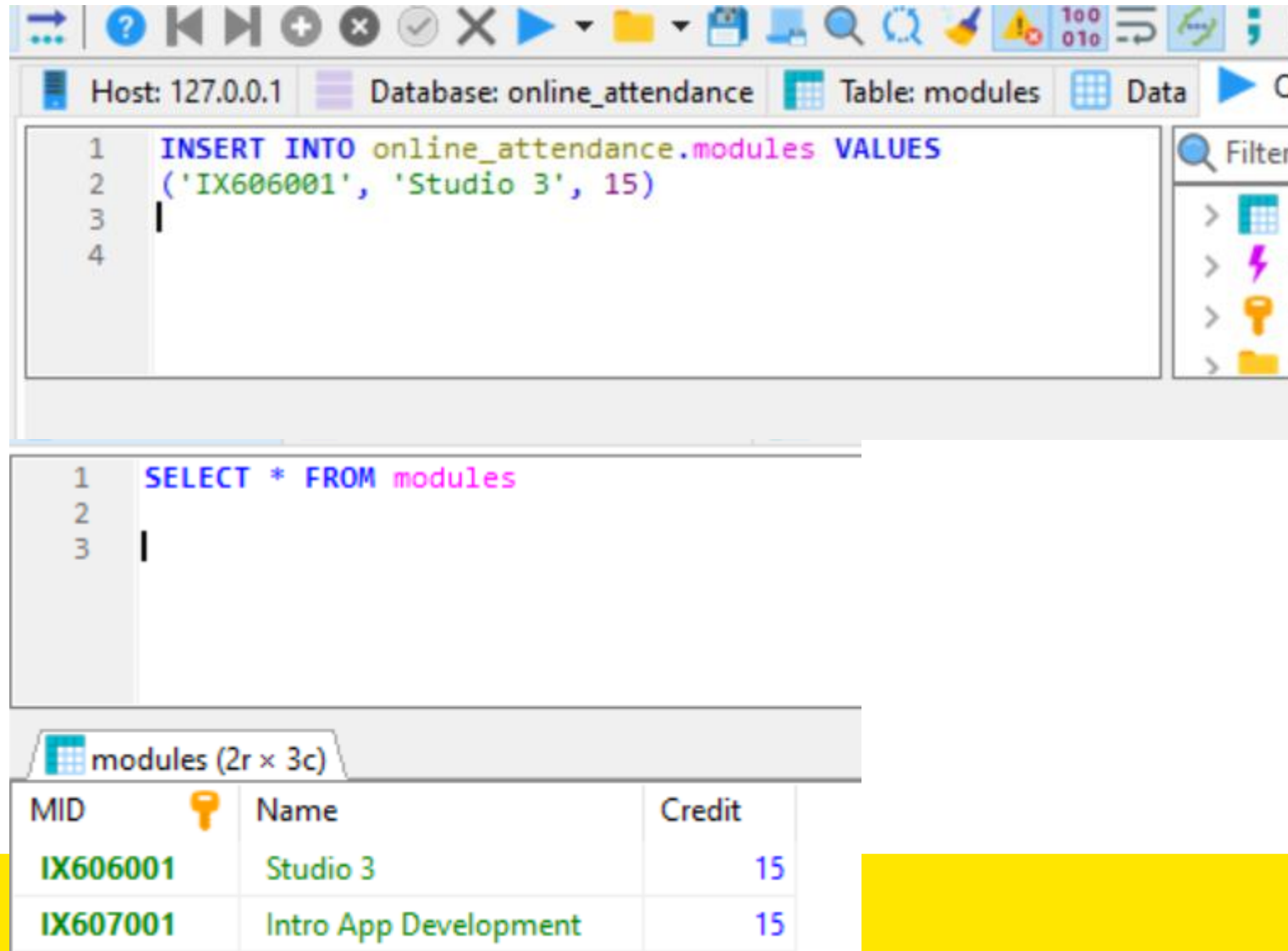
The screenshot shows a database query tool interface. At the top, there's a toolbar with various icons. Below it, a status bar displays 'Host: 127.0.0.1', 'Database: online_attendance', 'Table: modules', and 'Data'. The main query editor contains the following SQL query:

```
1 SELECT * FROM online_attendance.modules
```

The text 'online_attendance.modules' is highlighted in yellow. Below the query editor, a tab labeled 'modules (1r x 3c)' is active. The results are displayed in a table with three columns: MID, Name, and Credit.

MID	Name	Credit
IX607001	Intro App Development	15

Insert Data Using SQL



The screenshot shows a SQL IDE interface. The top toolbar contains various icons for navigation and execution. Below the toolbar, the status bar indicates the host is 127.0.0.1, the database is online_attendance, and the table is modules. The main editor area contains two SQL statements. The first statement is an INSERT INTO statement that adds a new row to the modules table. The second statement is a SELECT * FROM modules statement. Below the editor, a table view for the modules table is displayed, showing two rows of data.

```
1 INSERT INTO online_attendance.modules VALUES
2 ('IX606001', 'Studio 3', 15)
3
4
```

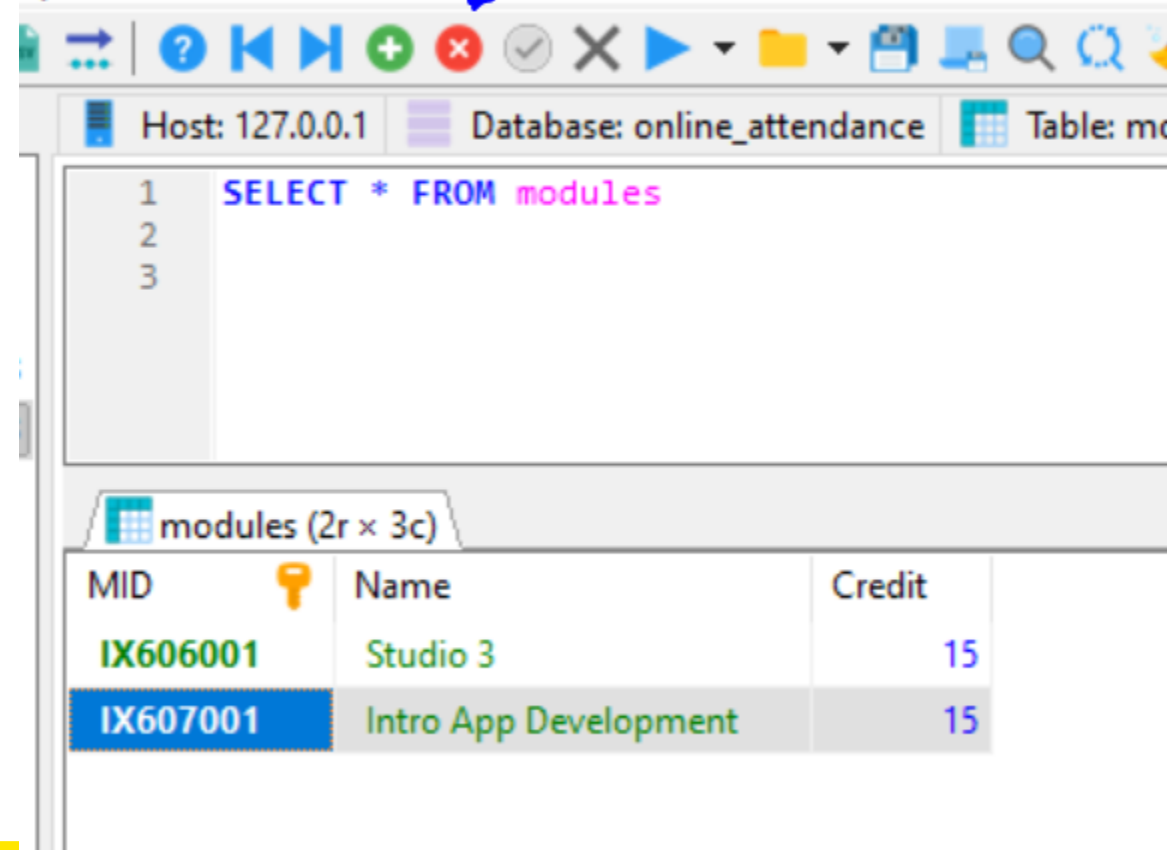
```
1 SELECT * FROM modules
2
3
```

MID	Name	Credit
IX606001	Studio 3	15
IX607001	Intro App Development	15

Delete a Row

diSQL Portable 11.2.0.6213

elp



The screenshot shows the diSQL Portable 11.2.0.6213 interface. A blue arrow points to the red 'X' icon in the toolbar, which is used for deleting rows. The toolbar also includes icons for help, back, forward, insert, delete, refresh, save, print, zoom in, and zoom out. Below the toolbar, the connection details are displayed: Host: 127.0.0.1, Database: online_attendance, and Table: modules. The SQL editor shows the query: `SELECT * FROM modules`. Below the editor, the results are displayed in a table with columns: MID, Name, and Credit. The table contains two rows: IX606001 (Studio 3) and IX607001 (Intro App Development). The row with MID IX607001 is highlighted in blue.

MID	Name	Credit
IX606001	Studio 3	15
IX607001	Intro App Development	15

Compound Primary Key

The screenshot shows a database management interface. On the left, a tree view shows the database structure with 'enrollments' selected under the 'online_attendance' database. On the right, the 'enrollments' table properties are shown, including a list of columns. A context menu is open over the columns, with 'Create new index' selected. Below the menu, a list of index types is shown: PRIMARY, KEY, and UNIQUE. A red arrow points to the 'Create new index' option in the context menu.

Select two columns

Columns:

#	Name	Datatype
1	MID	VARCHAR
2		
3		

Context Menu:

- Copy (Ctrl+C)
- Copy selected columns
- Paste columns
- Add column (Ctrl+Ins)
- Remove column (Ctrl+Del)
- Move up (Ctrl+U)
- Move down (Ctrl+D)
- Create new index
- Add to index

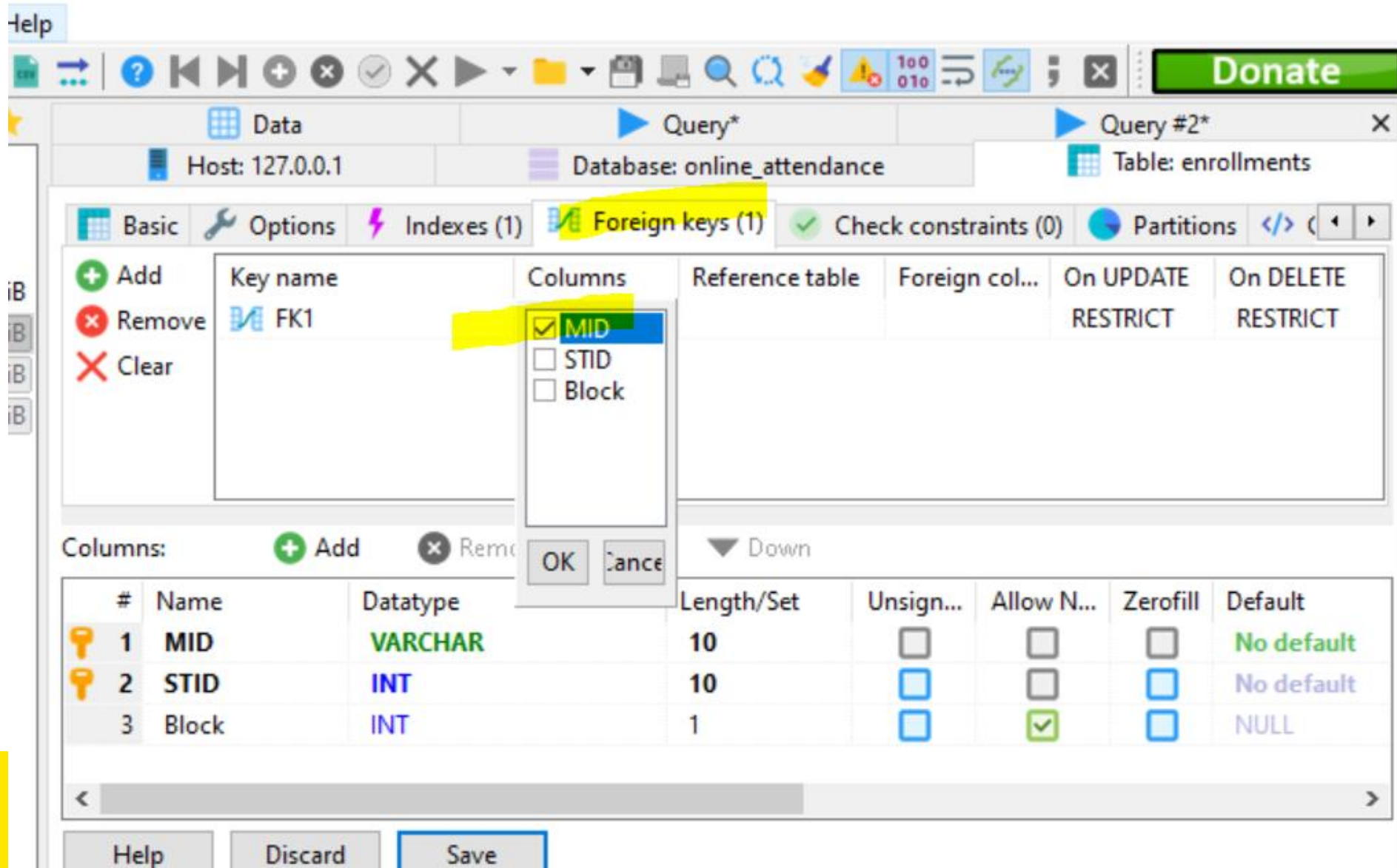
Index Types:

- PRIMARY
- KEY
- UNIQUE

SQL Query:

```
81 SHOW INDEXES FROM `enrollments` FROM
82 SELECT * FROM in
83 SELECT * FROM in
84 /* Entering sess
85 SHOW CREATE TABL
```


Foreign Keys



Foreign Keys

Host: 127.0.0.1 Database: online_attendance Table: enrollments

Basic Options Indexes (1) Foreign keys (1) Check constraints (0) Partitions

+ Add
x Remove
x Clear

Key name	Columns	Reference table	Foreign col...	On UPDATE	On DELETE
FK_enrollments_m...	MID	online_attend...	MID	RESTRICT	RESTRICT

Columns: + Add x Remove ▲ Up ▼ Down

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	MID	VARCHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
2	STID	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
3	Block	INT	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

< >

Help Discard Save

Foreign Keys

Database management interface showing foreign key configuration for the 'enrollments' table.

Host: 127.0.0.1 **Database:** online_attendance **Table:** enrollments

Foreign keys (2):

	Key name	Columns	Reference table	Foreign col...	On UPDATE	On DELETE
	FK_enrollments_m...	MID	online_attend...	MID	RESTRICT	RESTRICT
	FK_enrollments_stu...	STID	students	STID	RESTRICT	RESTRICT

Columns:

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	MID	VARCHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
2	STID	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
3	Block	INT	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Buttons: Help, Discard, **Save**

Export SQL Scripts

Laragon.MySQL\online_attendance\ - HeidiSQL Portable 12.1.0.6537

File Edit Search Query Tools Go to Help

Database filter Table filter Host: 127.0.0.1 Database: online_attendance enrollment.sql

Filter ...

- > Columns
- > SQL functions
- > SQL keywords
- > Snippets
- > Query history
- > Query profile
- > Bind parameters

Export database as SQL

Maintenance

Find text on server Shift+Ctrl+F

Bulk table editor

Expand all

Collapse all

Tree style options

Print... Ctrl+P

Refresh F5

Connection properties

Disconnect

```
1  --
2  -- Host: 127.0.0.1
3  -- Server version: 8.0.30 - MySQL Community Server - GPL
4  -- Server OS: Win64
5  -- HeidiSQL Version: 12.1.0.6537
6
7  --
8  -- @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
9  SET NAMES utf8 */;
10 SET NAMES utf8mb4 */;
11 -- @OLD_TIME_ZONE=@@TIME_ZONE */;
12 SET TIME_ZONE='+00:00' */;
13 -- @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
14 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
15 -- @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
```

355 SHOW PROCEDURE STATUS WHERE `Db`='online_attendance';

356 SHOW TRIGGERS FROM `online_attendance`;

357 SELECT *, EVENT_SCHEMA AS `Db`, EVENT_NAME AS `Name` FROM information_schema.`EVENTS` WHERE `EVENT_SCHEMA`='online_attendance';

358 USE `online_attendance`;

Dump database objects to an SQL file

Connected: 03:30 h MySQL 8.0.30 Uptime: 03:30 h Server time: 4:14 PM Idle.

Type here to search

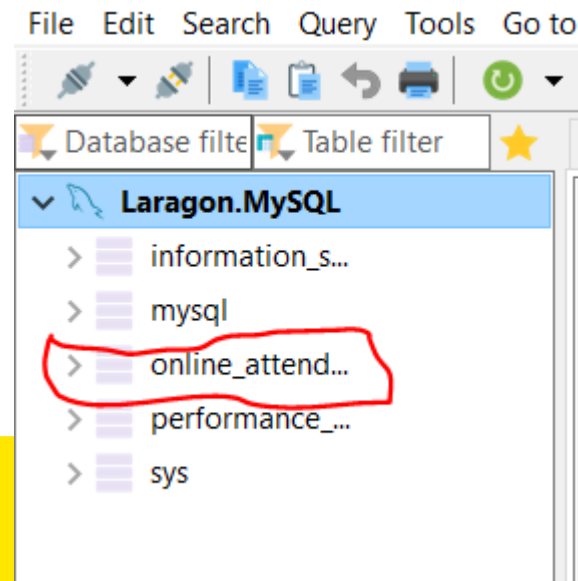
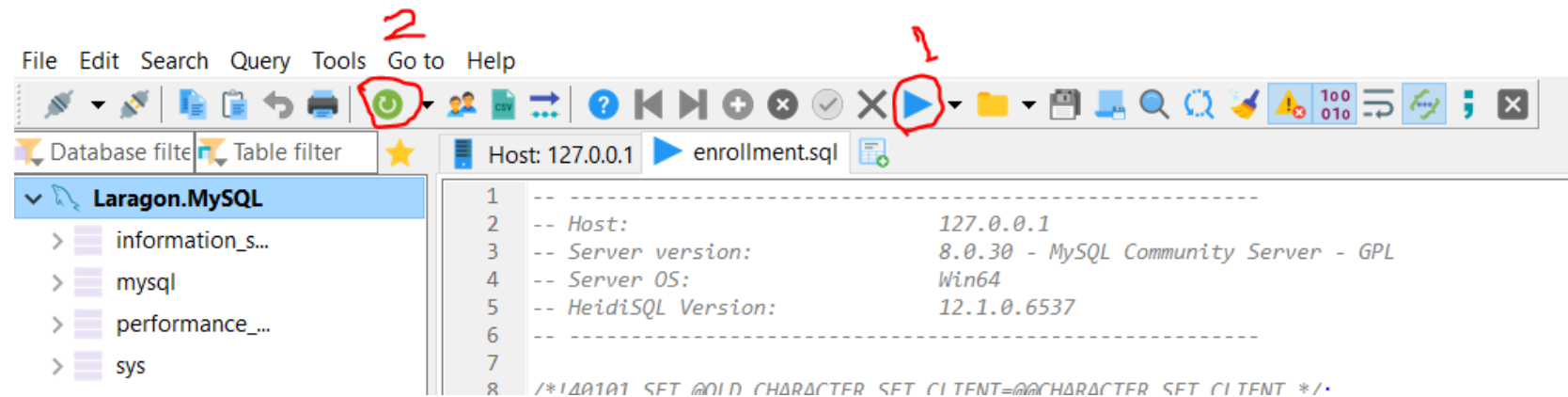
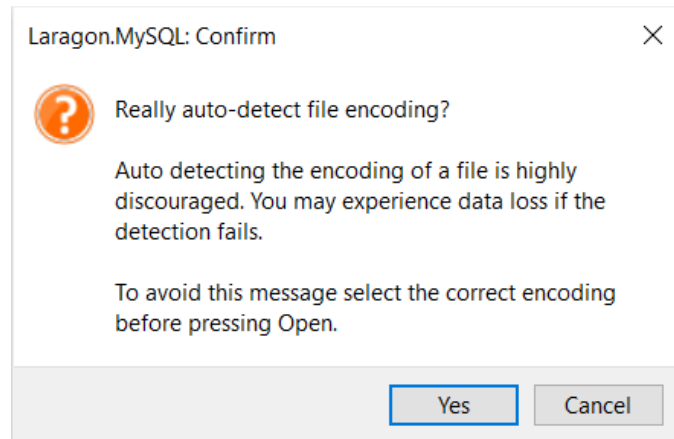
18°C 4:14 PM 23/04/2023

SQL Script

```
enrollment.sql X
enrollment.sql
1  -- -----
2  -- Host:                        127.0.0.1
3  -- Server version:             8.0.30 - MySQL Community Server - GPL
4  -- Server OS:                  Win64
5  -- HeidiSQL Version:           12.1.0.6537
6  -- -----
7
8  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
9  /*!40101 SET NAMES utf8 */;
10 /*!50503 SET NAMES utf8mb4 */;
11 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
14 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
15 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
16
17
18 -- Dumping database structure for enrollment
19 DROP DATABASE IF EXISTS `enrollment`;
20 CREATE DATABASE IF NOT EXISTS `enrollment` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80032 */;
21 USE `enrollment`;
22
23 -- Dumping structure for table enrollment.enrollment
24 DROP TABLE IF EXISTS `enrollment`;
25 CREATE TABLE IF NOT EXISTS `enrollment` (
26   `STID` int NOT NULL,
27   `MID` varchar(11) NOT NULL,
28   `LID` int DEFAULT NULL,
29   `Block` int NOT NULL,
30   `Mark` float(3,3) DEFAULT NULL,
31   PRIMARY KEY (`STID`,`MID`,`Block`)
32 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
33
34 -- Dumping data for table enrollment.enrollment: ~2 rows (approximately)
35 REPLACE INTO `enrollment` (`STID`,`MID`,`LID`,`Block`,`Mark`) VALUES
36   (192833, 'IX606001', 200, 5, NULL),
37   (192833, 'IX607001', 100, 5, NULL);
```

Importing SQL Scripts

File -> Load SQL File



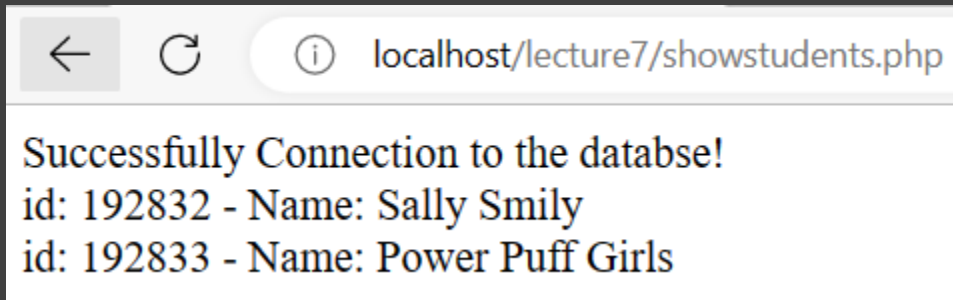
PHP – Connect to Database

```
<?php
//connectDB.php
$servername = "localhost";
$username = "root";
$password = ""; //Your password here
$dbname = "online_attendance";
?>
```

```
<?php
//testDB.php
require_once 'connectDB.php';
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error)
    die("Connection failed: " . $conn->connect_error);
else
    echo "Successfully Connection to the database!";
$conn->close();
?>
```

PHP – Querying the Database

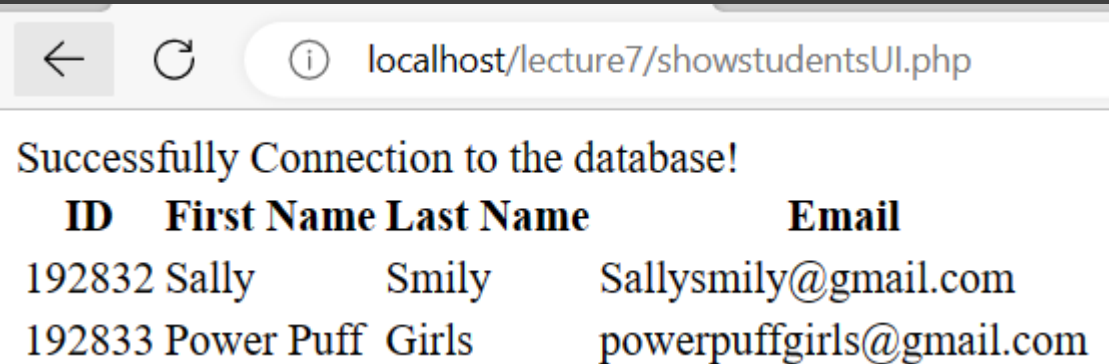
```
<?php
require_once 'connectDB.php';
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error)
    die("Connection failed: " . $conn->connect_error);
else
    echo "Successfully Connection to the database!". "<br>";
$sql = "SELECT * FROM students";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["STID"]. " - Name: " . $row["Name"]. " " . $row["Lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```



PHP – Querying Data - HTML Output

```
$sql = "SELECT * FROM students" ;
$stmt = $conn->prepare($sql);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows > 0) {
    ?>
    <table >
        <thead>
            <tr>
                <th> ID </th>
                <th>First Name</th>
                <th>Last Name</th>
                <th>Email </th>
            </tr>
        </thead>
        <tbody>
            <?php
            // output data of each row
```

```
while($row = $result->fetch_assoc()) {?>
    <tr>
        <td><?php echo $row["STID"]; ?></td>
        <td><?php echo $row["Name"]; ?></td>
        <td><?php echo $row["Lastname"]; ?></td>
        <td><?php echo $row["Email"]; ?></td>
    </tr>
    <?php
    }
} else {
    echo "0 results";
}
```



← ↻ ⓘ localhost/lecture7/showstudentsUI.php

Successfully Connection to the database!

ID	First Name	Last Name	Email
192832	Sally	Smily	Sallysmily@gmail.com
192833	Power Puff	Girls	powerpuffgirls@gmail.com

PHP – Insert Data

```
$sql = "INSERT INTO students (STID, Name, Lastname, Email)
VALUES (192332, 'John', 'Doe', 'john@example.com');"
```

```
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
```

localhost/lecture7/addstudent.php

Successfully Connection to the databse!
New record created successfully

File Edit Search Query Tools Go to Help

Database filter Table filter Host: 127.0.0.1 Database: online_attendance Table: students Data enrollment

Laragon.MySQL

- information_s...
- mysql
- online_atten... 64.0 KiB
 - enrollment 16.0 KiB
 - lecturer 16.0 KiB
 - module 16.0 KiB
 - students 16.0 KiB

online_attendance.students: 3 rows total (approximately)

STID	Name	Lastname	Email	Address
192,332	John	Doe	john@example.com	(NULL)
192,832	Sally	Smily	Sallysmily@gmail.com	458 Kings Road
192,833	Power Puff	Girls	powerpuffgirls@gmail.com	458 Mojojojo Road

PHP – Edit Data

```
$sql = "UPDATE students SET Name = 'Ali' WHERE STID=192332";  
if ($conn->query($sql) === TRUE) {  
    echo "Record updated successfully";  
} else {  
    echo "Error updating record: " . $conn->error;  
}
```

localhost/lecture7/editstudent.php

Successfully Connection to the databse!
Record updated successfully

File Edit Search Query Tools Go to Help

Database filter Table filter Host: 127.0.0.1 Database: online_attendance Table: students Data en

Laragon.MySQL

- information_s...
- mysql
- online_atten... 64.0 KiB
 - enrollment 16.0 KiB
 - lecturer 16.0 KiB
 - module 16.0 KiB
 - students** 16.0 KiB

online_attendance.students: 3 rows total (approximately)

STID	Name	Lastname	Email	Address
192,332	Ali	Doe	john@example.com	(NULL)
192,832	Sally	Smily	Sallysmily@gmail.com	458 Kings Road
192,833	Power Puff	Girls	powerpuffgirls@gmail.com	458 Mojojojo Road

PHP – Delete Data

```
$sql = "DELETE FROM students WHERE STID=192332";  
if ($conn->query($sql) === TRUE) {  
    echo "Record deleted successfully";  
} else {  
    echo "Error deleting record: " . $conn->error;  
}
```

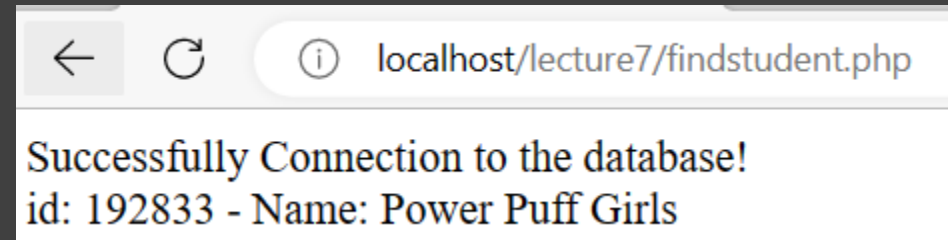
PHP – Create a Table

```
$sql = "CREATE TABLE enrollments_new (  
    STID INT NOT NULL,  
    MID VARCHAR(10) NOT NULL,  
    Block INT,  
    PRIMARY KEY (STID,MID),  
    FOREIGN KEY (STID) REFERENCES students(STID),  
    FOREIGN KEY (MID) REFERENCES modules(MID)  
)";  
$conn->query($sql);  
echo "Database and table users created successfully.";
```

PHP – Passing Parameter to SQL

```
$id = 192833;
$sql = "SELECT * FROM students WHERE STID=?"; // SQL with parameters
$stmt = $conn->prepare($sql);
$stmt->bind_param("i", $id);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["STID"]. " - Name: " . $row["Name"]. " " . $row["Lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
```



? in the SQL query shows a parameter which must be set by using bind_param method. The first argument is a string which shows the data type:

- i for integer
- d for double (float)
- s for string
- b for blob

In this example, since the student ID is an integer, we used i.

PHP – Passing Parameter to SQL – Cont.

```
$id = 1923443;
$name= "Jack";
$sql = "SELECT * FROM students WHERE STID=? and Name=?"; // SQL with parameters
$stmt = $conn->prepare($sql);
$stmt->bind_param("is", $id, $name);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["STID"]. " - Name: " . $row["Name"]. " " . $row["Lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
```

The number of parameters in bind_param must match with ? in SQL query (two here). The first parameter, “is”, shows the first parameter is an integer (id) and the second one is a string (name).

Passing Form Variables to SQL

```
<html>
<body>

<form action="addStudentFromForm.php"
method="post">
    Student ID: <input type="text"
name="STID"><br>
    Name: <input type="text" name="name"><br>
    Last name: <input type="text"
name="lname"><br>
    E-mail: <input type="text" name="email"><br>
    Address: <input type="text"
name="address"><br>
    <input type="submit">
</form>
```

```
</body>
```

```
</html>
```

If you add the following statement to the end of addstudentfromform.php, you will be forwarded to studentform.php

```
header("Location:studentform.php");
```

```
<?php
// addStudentFromForm.php
require_once 'connectDB.php';
$id = $_POST["STID"];
$name = $_POST["name"];
$lname = $_POST["lname"];
$email = $_POST["email"];
$address = $_POST["address"];

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error)
    die("Connection failed: " . $conn->connect_error);
$sql = "INSERT INTO students (STID, Name, Lastname, Email, Address)
VALUES (?, ?, ?, ?, ?)";
if ($stmt = $conn->prepare($sql))
    $stmt->bind_param("issss", $id, $name, $lname, $email, $address);
else
{
    $error = $conn->errno . ' ' . $conn->error;
    echo $error;
}
$stmt->execute();
echo "Student has been successfully added!";
$conn->close();
?>
```

Using PHP Variables in JS Code

```
$sql = "SELECT * FROM students";
$result = $conn->query($sql);
$i = 0;
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        $rows[$i] = $row;
        $i++;
    }
} else {
    echo "0 results";
}
$conn->close();
?>
<HTML>
<script>
    var result = <?php echo json_encode($rows); ?>;
    for(i=0; i < result.length; i++)
        alert(result[i]["STID"]+" "+result[i]["Name"]+" "+result[i]["Lastname"]+" "+result[i]["Email"]+"
"+result[i]["Address"]);
</Script>
</HTML>
```

JS Variable

PHP Variable

Connecting Backend to Frontend

Demo – Student List

Final Note

- **Only one lab this week:** Wednesday 26th April.
- **Marks for Project 1 have been released.**
 - Please check Moodle.
 - Let us know if you have any question about the feedback or marks.
- **Project 2 assessment is now open.**
 - Task Description is now available on Moodle under: Assessment Hub → Project 2.
- **Please do not forget Quiz 2 is next week:**
 - Quiz will be during your scheduled lab time.
 - Quiz is closed book.
 - Quiz duration is 40 minutes.
 - Quiz will cover material discussed in weeks 4, 5, and 6.