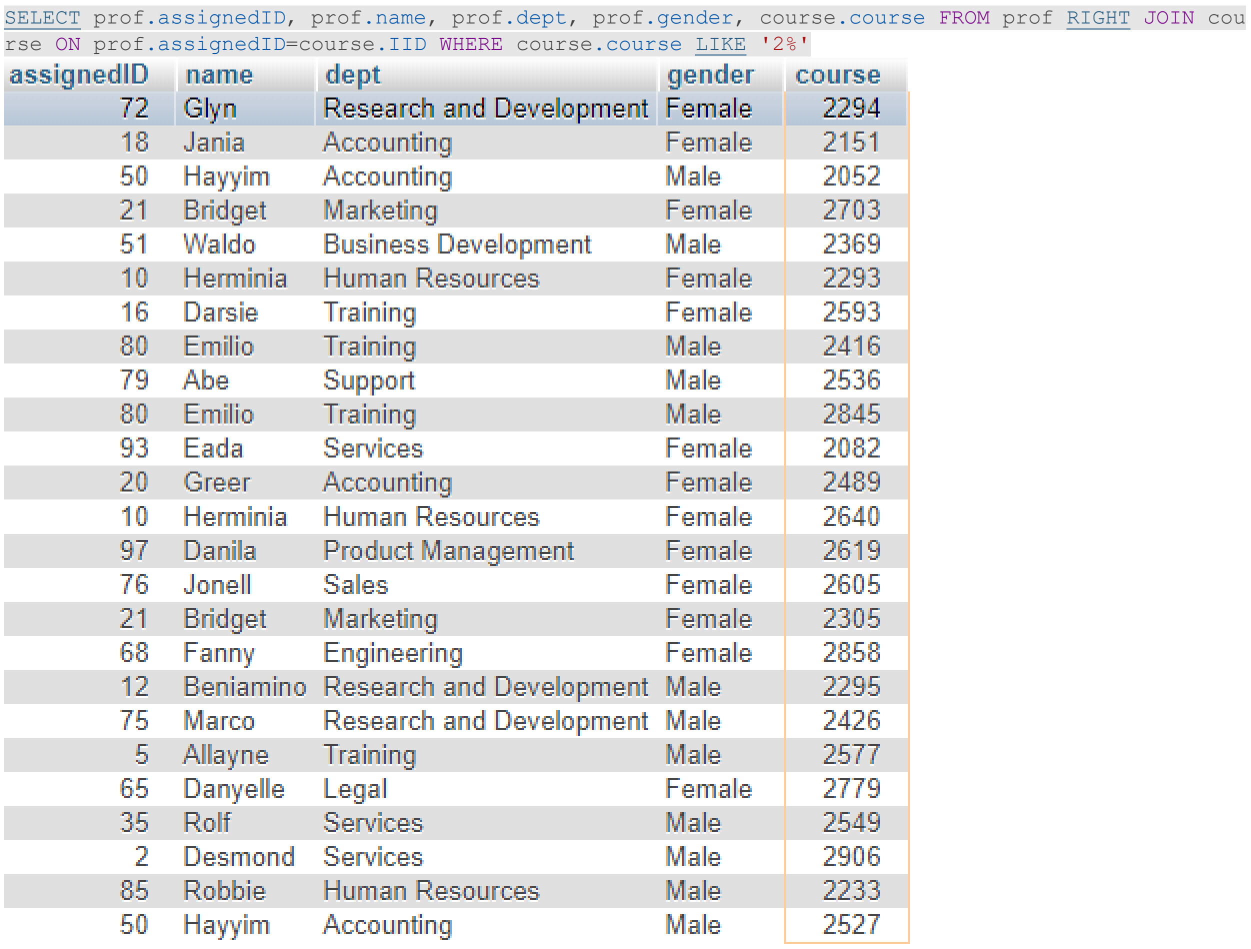
Part 1:

1 - Construct an SQL query that will show the assignedID, professorName, and the professor's home department for each professor that teaches a 2000 level course (all course codes that begin with a two). (2 points)

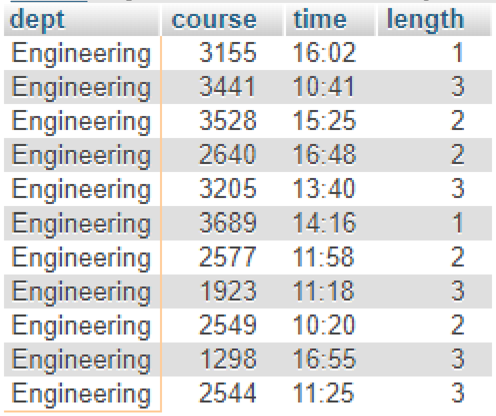


```
SELECT prof.assignedID, prof.name, prof.dept, prof.gender, course.course FROM prof RIGHT JOIN cou
rse ON prof.assignedID=course.IID WHERE course.course LIKE '2%'
```

| assignedID | name | dept | gender | course |
|---|---|---|---|---|
| 72 | Glyn | Research and Development | Female | 2294 |
| 18 | Jania | Accounting | Female | 2151 |
| 50 | Hayyim | Accounting | Male | 2052 |
| 21 | Bridget | Marketing | Female | 2703 |
| 51 | Waldo | Business Development | Male | 2369 |
| 10 | Herminia | Human Resources | Female | 2293 |
| 16 | Darsie | Training | Female | 2593 |
| 80 | Emilio | Training | Male | 2416 |
| 79 | Abe | Support | Male | 2536 |
| 80 | Emilio | Training | Male | 2845 |
| 93 | Eada | Services | Female | 2082 |
| 20 | Greer | Accounting | Female | 2489 |
| 10 | Herminia | Human Resources | Female | 2640 |
| 97 | Danila | Product Management | Female | 2619 |
| 76 | Jonell | Sales | Female | 2605 |
| 21 | Bridget | Marketing | Female | 2305 |
| 68 | Fanny | Engineering | Female | 2858 |
| 12 | Beniamino | Research and Development | Male | 2295 |
| 75 | Marco | Research and Development | Male | 2426 |
| 5 | Allayne | Training | Male | 2577 |
| 65 | Danyelle | Legal | Female | 2779 |
| 35 | Rolf | Services | Male | 2549 |
| 2 | Desmond | Services | Male | 2906 |
| 85 | Robbie | Human Resources | Male | 2233 |
| 50 | Hayyim | Accounting | Male | 2527 |

2 - Construct an SQL query that will show all the course IDs, lectureTimes, and Lecture durations for the Engineering department. (2 points)

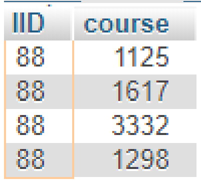
```
SELECT dept, course, `time`, length FROM course WHERE (dept = 'Engineering')
```



| dept | course | time | length |
|---|---|---|---|
| Engineering | 3155 | 16:02 | 1 |
| Engineering | 3441 | 10:41 | 3 |
| Engineering | 3528 | 15:25 | 2 |
| Engineering | 2640 | 16:48 | 2 |
| Engineering | 3205 | 13:40 | 3 |
| Engineering | 3689 | 14:16 | 1 |
| Engineering | 2577 | 11:58 | 2 |
| Engineering | 1923 | 11:18 | 3 |
| Engineering | 2549 | 10:20 | 2 |
| Engineering | 1298 | 16:55 | 3 |
| Engineering | 2544 | 11:25 | 3 |

3 - Business Development 3332 is no longer being taught by George (assignedID = 77), it is being taught by Royall (assignedID = 88). Update the course table to reflect this change. (3 points)

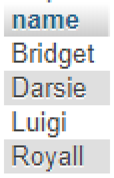
```
UPDATE course SET IID = 88 WHERE course = 3332
```



| IID | course |
|---|---|
| 88 | 1125 |
| 88 | 1617 |
| 88 | 3332 |
| 88 | 1298 |

4 - In this imaginary university, they have a rule that states any professor cannot lecture for more than 6 hours per week. Construct an SQL query that will find any professor that is lecturing for more than 6 hours per week (please note the result set may be empty). (3 points)

```
SELECT prof.name FROM (course RIGHT JOIN prof ON prof.assignedID=course.IID) GROUP BY name HAVING
 SUM(length) > 6
```

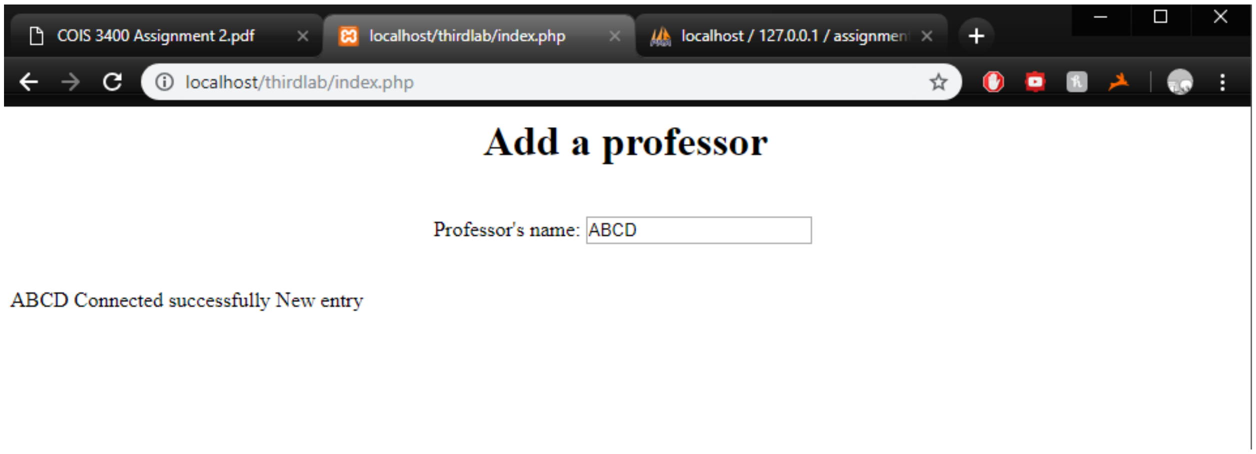


| name |
|---|
| Bridget |
| Darsie |
| Luigi |
| Royall |

5 Bonus: write a PHP or Python DB-API script with an HTML5 frontend that will add a professor to the professors table. (2 points)

```
<html>
	<body>
		<center>
			<H1> Add a professor </H1> <br>

			<form action="index.php" method="post">
				Professor's name: <input type="text" name="name">     <br> <br>

		</center>
			</form>
	</body>
</html>

<?php   // PHP tag

$name = $_POST['name'];          // Collect input

echo($name);    // Debug input

$conn = mysqli_connect("localhost","root","","assignment2");    // Connect to a2 database

// Check for connection failures
if (mysqli_connect_errno($conn)){
	echo " Failed to connect: ".mysqli_connect_error();
} else {
	echo " Connected successfully\n";
}

$sql = "INSERT INTO prof(`name`) VALUES ('$name')";     // SQL statement to insert name

// Check for query failures
if ($conn->query($sql) === TRUE) {
	echo "New entry";
} else {
	echo "Error " . $sql . $conn->error;
}

mysqli_close($conn);     // Close connection

?>
```

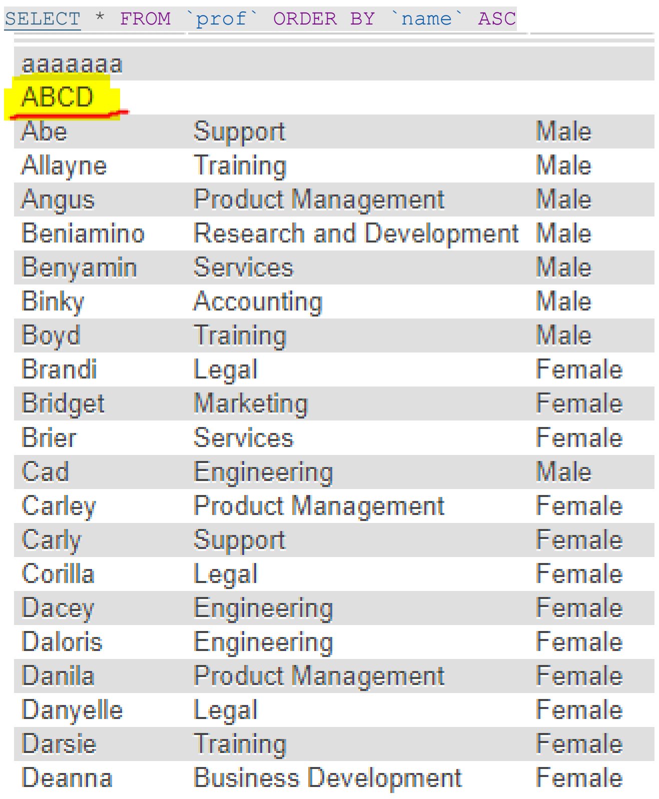
Example:


COIS 3400 Assignment 2.pdf
localhost/thirdlab/index.php
localhost / 127.0.0.1 / assignmen
localhost/thirdlab/index.php

# Add a professor

Professor's name: ABCD

ABCD Connected successfully New entry

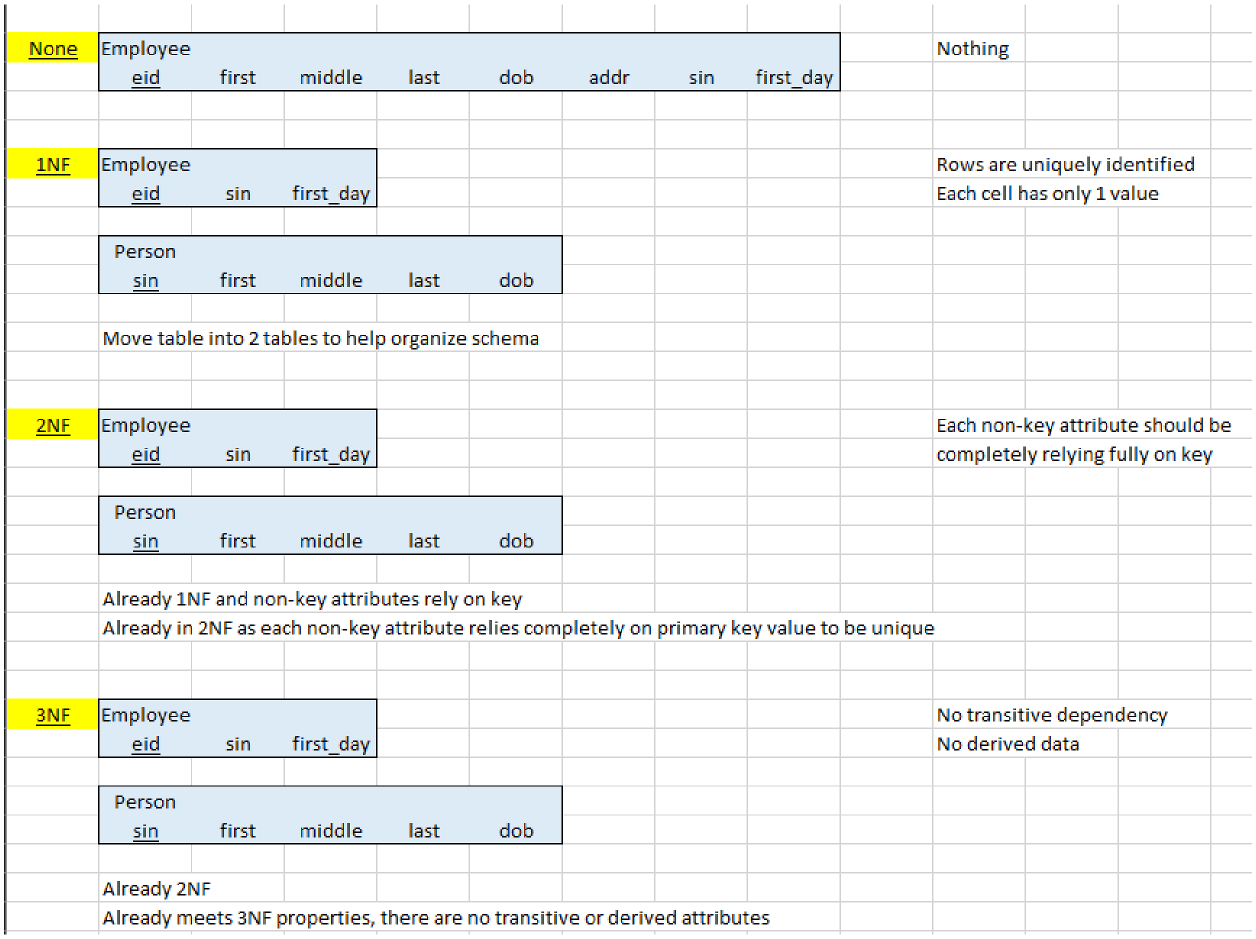


```
SELECT * FROM `prof` ORDER BY `name` ASC
```

| | | |
|---|---|---|
| aaaaaaa | | |
| ABCD | | |
| Abe | Support | Male |
| Allayne | Training | Male |
| Angus | Product Management | Male |
| Beniamino | Research and Development | Male |
| Benyamin | Services | Male |
| Binky | Accounting | Male |
| Boyd | Training | Male |
| Brandi | Legal | Female |
| Bridget | Marketing | Female |
| Brier | Services | Female |
| Cad | Engineering | Male |
| Carley | Product Management | Female |
| Carly | Support | Female |
| Corilla | Legal | Female |
| Dacey | Engineering | Female |
| Daloris | Engineering | Female |
| Danila | Product Management | Female |
| Danyelle | Legal | Female |
| Darsie | Training | Female |
| Deanna | Business Development | Female |

Part 2

1 - Normalize the following DB schema up until 3rd normal form: Employee(eid, first name, middle name, last name, date_of_birth, home_address, national_insurance_number, first_day_of_employment). Remember you need to show your steps starting with 1NF, 2NF, until 3NF. (5 points)



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| None | Employee | | | | | | | | Nothing |
| | eid | first | middle | last | dob | addr | sin | first_day | |
| | | | | | | | | | |
| 1NF | Employee | | | | | | | | Rows are uniquely identified |
| | eid | sin | first_day | | | | | | Each cell has only 1 value |
| | | | | | | | | | |
| | Person | | | | | | | | |
| | sin | first | middle | last | dob | | | | |
| | | | | | | | | | |
| | Move table into 2 tables to help organize schema | | | | | | | | |
| | | | | | | | | | |
| 2NF | Employee | | | | | | | | Each non-key attribute should be |
| | eid | sin | first_day | | | | | | completely relying fully on key |
| | | | | | | | | | |
| | Person | | | | | | | | |
| | sin | first | middle | last | dob | | | | |
| | | | | | | | | | |
| | Already 1NF and non-key attributes rely on key | | | | | | | | |
| | Already in 2NF as each non-key attribute relies completely on primary key value to be unique | | | | | | | | |
| | | | | | | | | | |
| 3NF | Employee | | | | | | | | No transitive dependency |
| | eid | sin | first_day | | | | | | No derived data |
| | | | | | | | | | |
| | Person | | | | | | | | |
| | sin | first | middle | last | dob | | | | |
| | | | | | | | | | |
| | Already 2NF | | | | | | | | |
| | Already meets 3NF properties, there are no transitive or derived attributes | | | | | | | | |