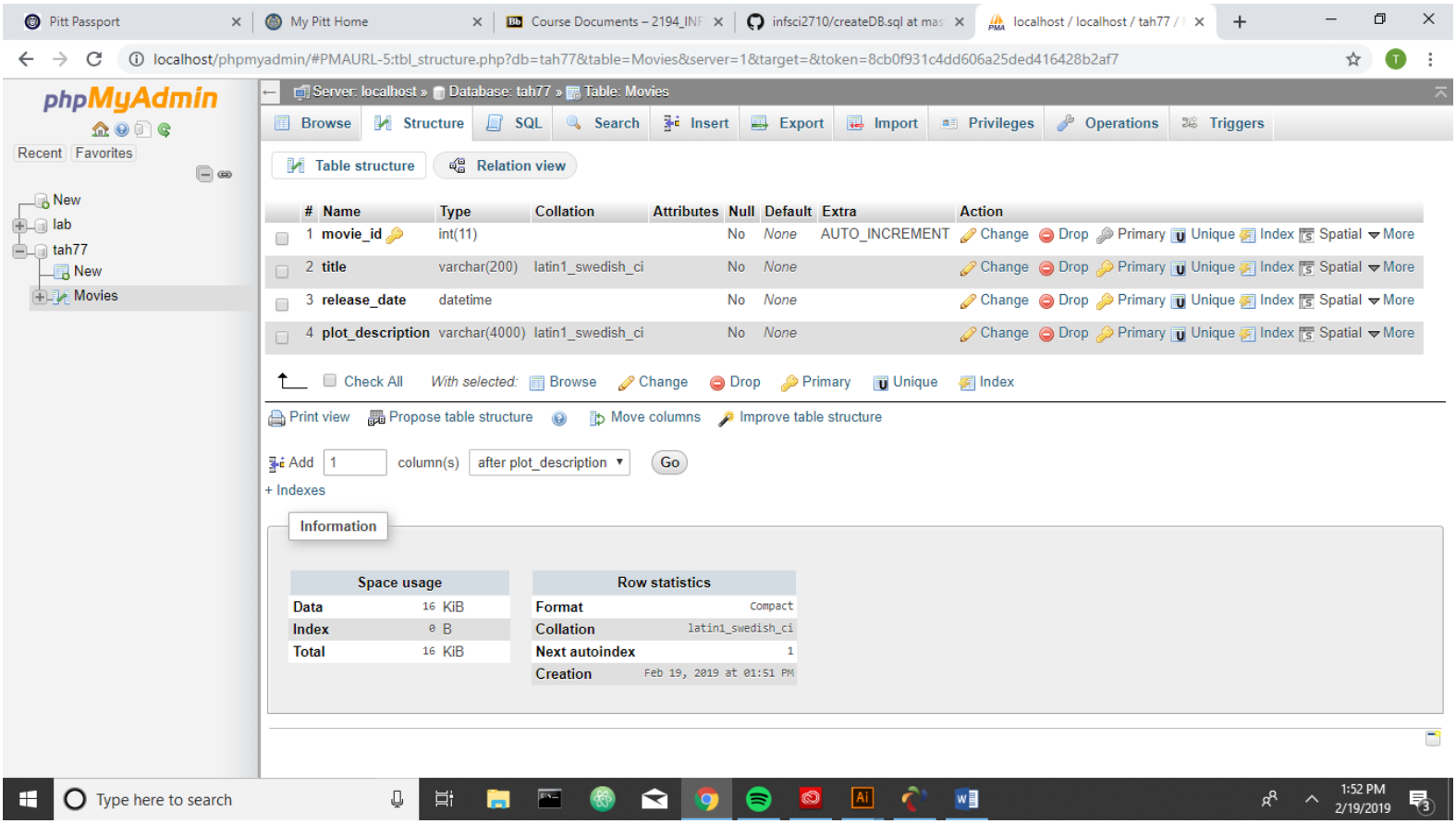
Taylor Herb
Assignment 2

Task 1 (15 points): In database **[your Pitt username]**, create the following entity tables:

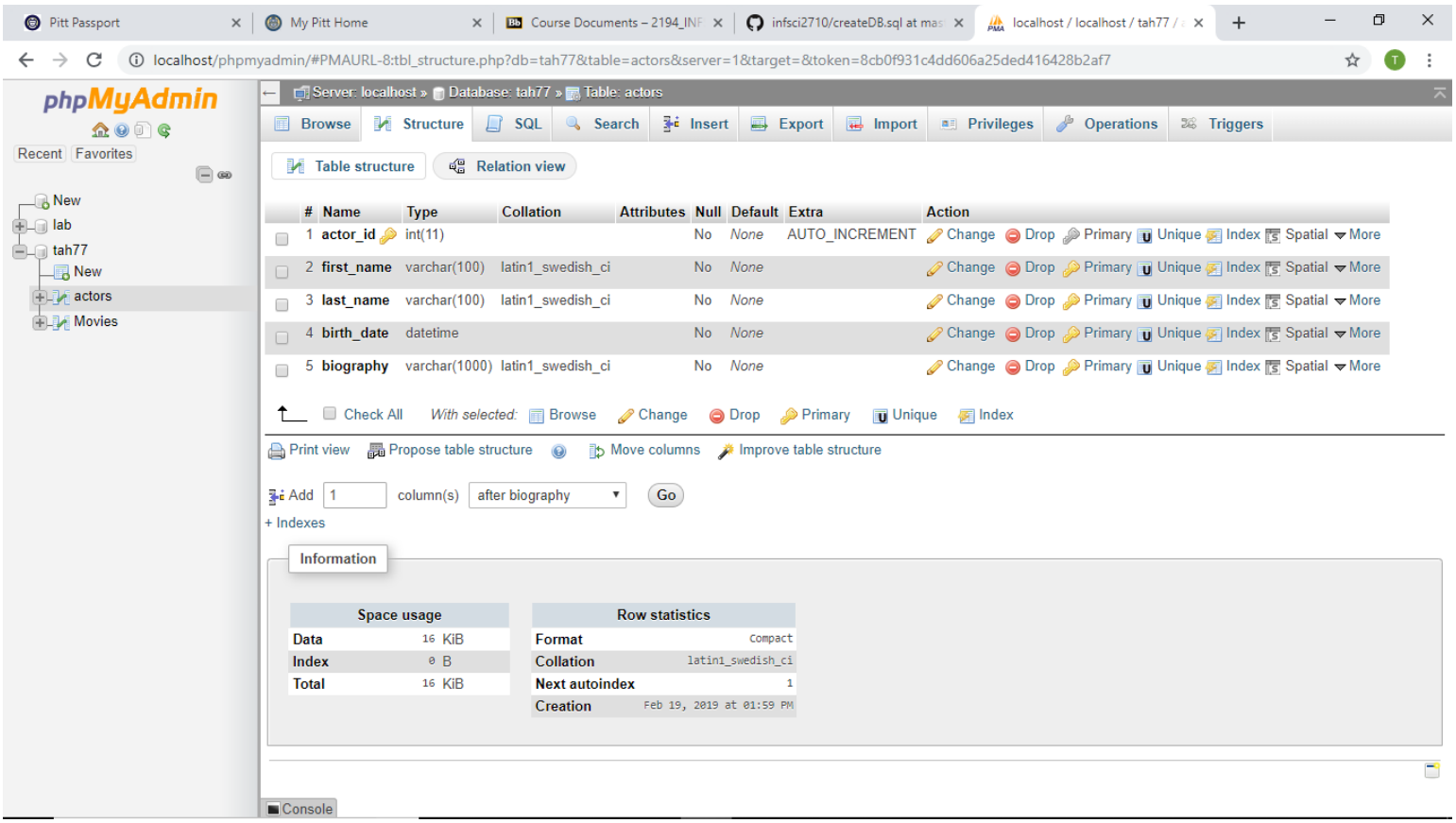
1. Movies

```
CREATE TABLE `tah77`.`Movies` ( `movie_id` INT NOT NULL AUTO_INCREMENT ,`title` VARCHAR(200) NOT NULL , `release_date` DATETIME NOT NULL ,`plot_description` VARCHAR(4000) NOT NULL , PRIMARY KEY (`movie_id`)) ENGINE =InnoDB;
```



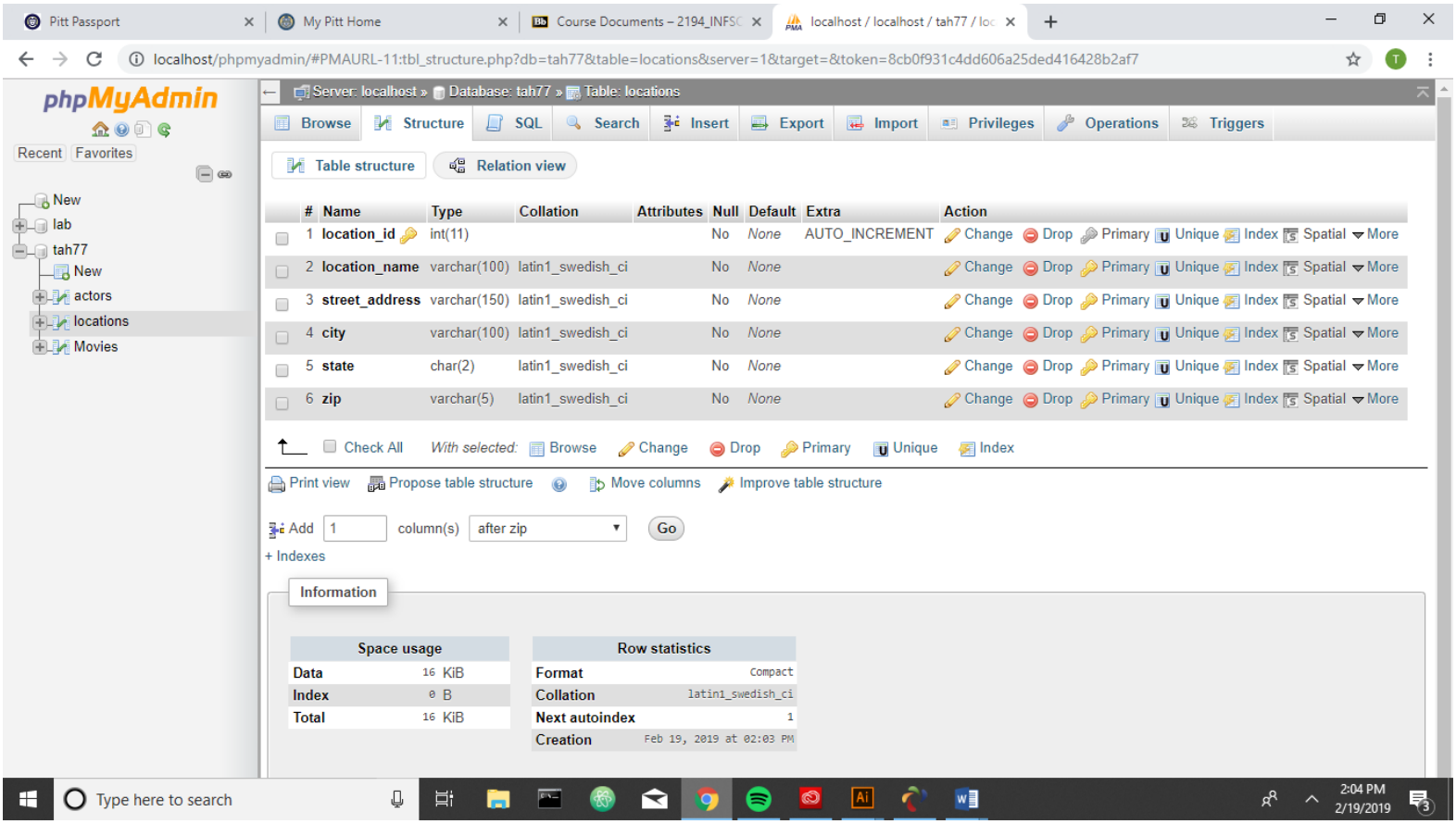
2. Actors

```
CREATE TABLE `tah77`.`actors` ( `actor_id` INT NOT NULL AUTO_INCREMENT ,`first_name` VARCHAR(100) NOT NULL , `last_name` VARCHAR(100) NOT NULL ,`birth_date` DATETIME NOT NULL , `biography` VARCHAR(1000) NOT NULL , PRIMARYKEY (`actor_id`)) ENGINE = InnoDB;
```



3. Locations

```
CREATE TABLE `tah77`.`locations` ( `location_id` INT NOT NULL AUTO_INCREMENT ,`loc
ation_name` VARCHAR(100) NOT NULL , `street_address` VARCHAR(150) NOT NULL, `city`
 VARCHAR(100) NOT NULL , `state` CHAR(2) NOT NULL , `zip` VARCHAR(5)NOT NULL , PRI
MARY KEY (`location_id`)) ENGINE = InnoDB;
```



Each table's logical structure should correspond to the descriptions provided in this assignment. Use CREATE TABLE statement.

Task 2 (10 points): In your database, create the following junction tables:

1. movies_actors

Create table:

```
CREATE TABLE `tah77`.`movies_actors` ( `actor_id` INT NOT NULL , `movie_id`INT NOT NULL )
 ENGINE = InnoDB;
```

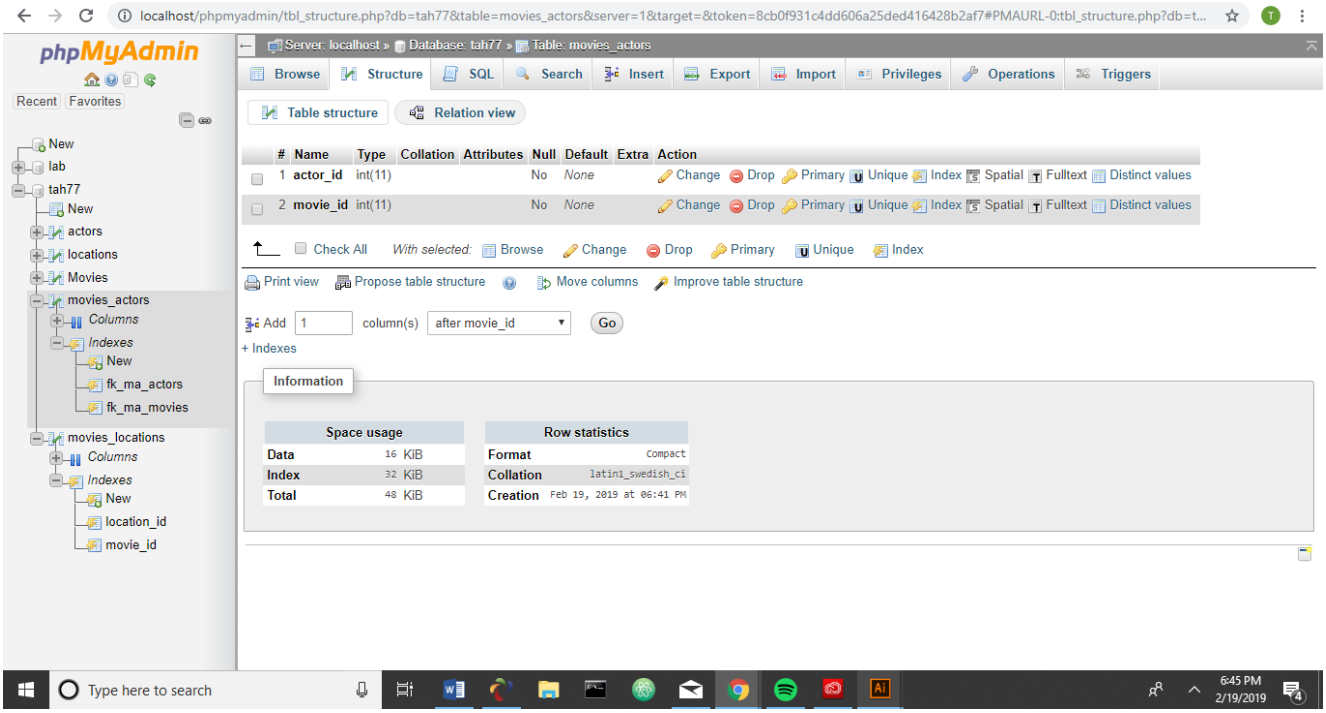
Add movie_id foreign key:

```
ALTER TABLE movies_actors
ADD CONSTRAINT `fk_ma_movies`
FOREIGN KEY (`movie_id`) REFERENCES `Movies`(`movie_id`);
```

Add actor_id foreign key:

```
ALTER TABLE movies_actors
ADD CONSTRAINT `fk_ma_actors`
FOREIGN KEY (`movie_actors`) REFERENCES `actors`(`actor_id`);
```

*deleted & edited after original creation to add constraint 'fk_' names



2.

movies_locations

Create table:

```
CREATE TABLE `tah77`.`movies_locations` ( `movie_id` INT NOT NULL ,`location_id` INT NOT NULL ) ENGINE = InnoDB;
```

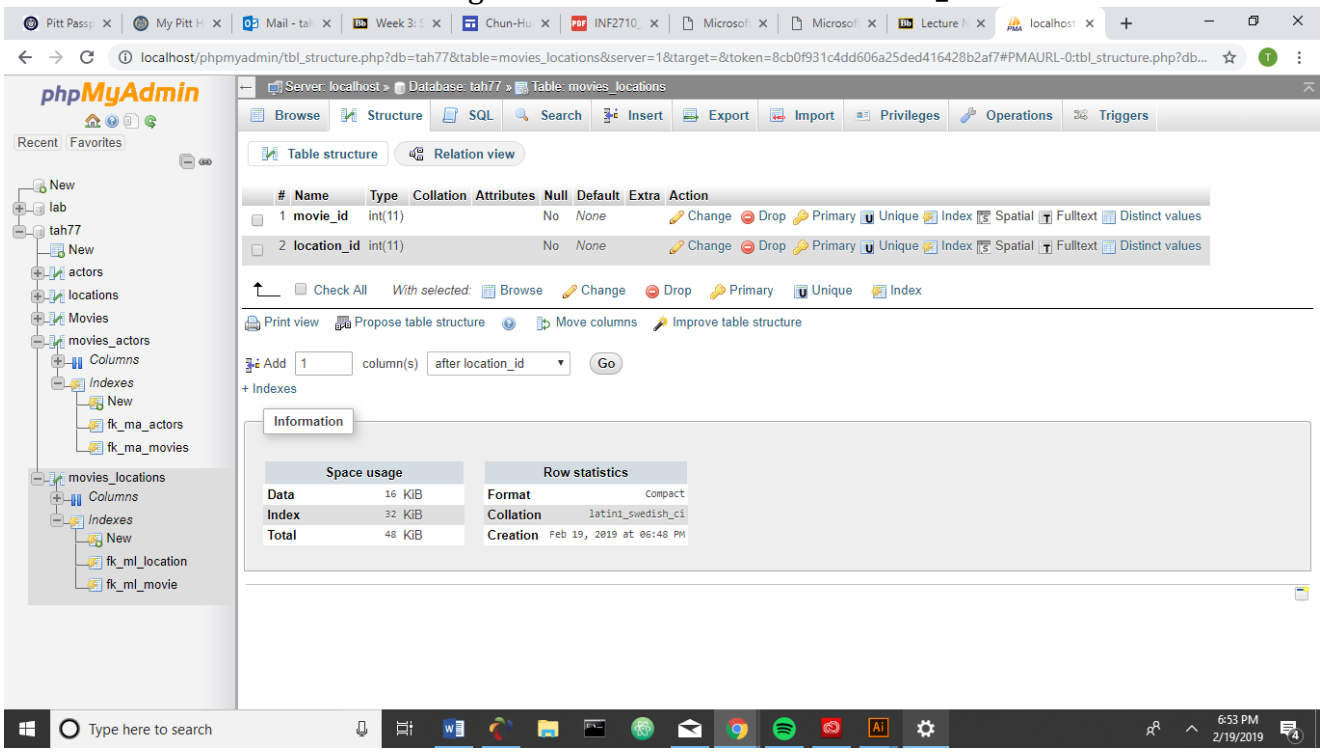
Add movie_id foreign key:

```
ALTER TABLE movies_locations
ADD CONSTRAINT `fk_ml_movies`
FOREIGN KEY (`moviesid`) REFERENCES `Movies`(`movie_id`);
```

Add location_id foreign key:

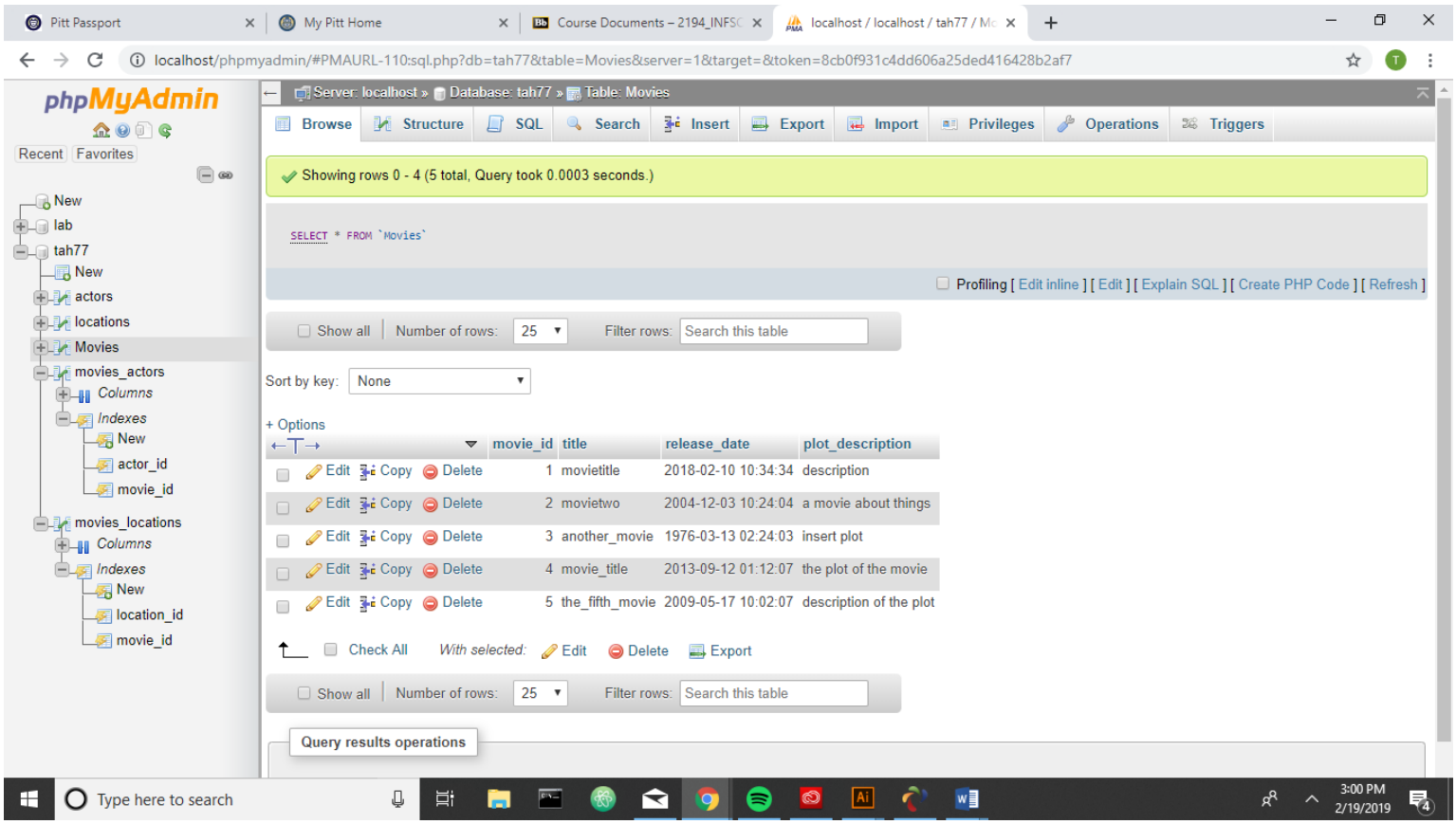
```
ALTER TABLE movies_locations
ADD CONSTRAINT `fk_ml_location`
FOREIGN KEY (`location_id`) REFERENCES `locations`(`location_id`);
```

*deleted & edited after original creation to add constraint ‘fk_’ names



Use CREATE TABLE statement to create junction tables. Make sure to create appropriate foreign keys – each table will have two foreign keys. Use ALTER TABLE statement to create foreign keys.

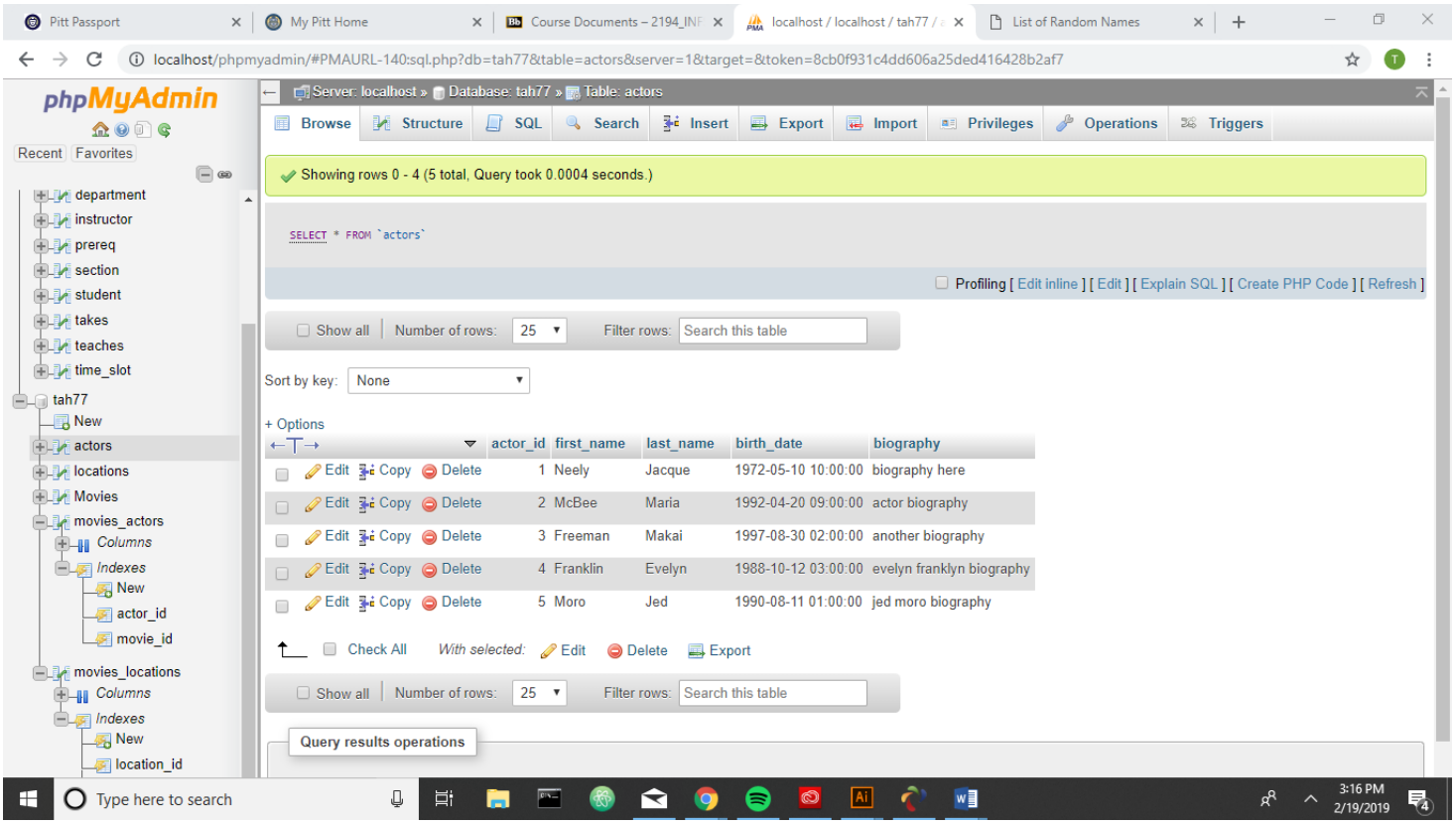
Task 3 (15 points): For each entity table, insert at least 5 rows using INSERT statement:
At least 5 movies in the **movies** table
insert into Movies values ('1', 'movietitle', ' 2018-02-10 10:34:34', 'description');
insert into Movies values ('2', 'movietwo', ' 2004-12-03 10:24:04', 'a movie about things');
insert into Movies values ('3', 'another_movie', ' 1976-3-13 02:24:03', 'insert plot');
insert into Movies values ('4', 'movie_title', ' 2013-9-12 01:12:07', 'the plot of the movie');
insert into Movies values ('5', 'the_fifth_movie', ' 2009-05-17 10:02:07', 'description of the plot');



1. At least 5 actors in the **actors** table

insert into actors values ('1', 'Neely', 'Jacque', '1972-05-10 10:00:00', 'biography here');
insert into actors values ('2', 'McBee', 'Maria', '1992-04-20 09:00:00', 'actor biography');

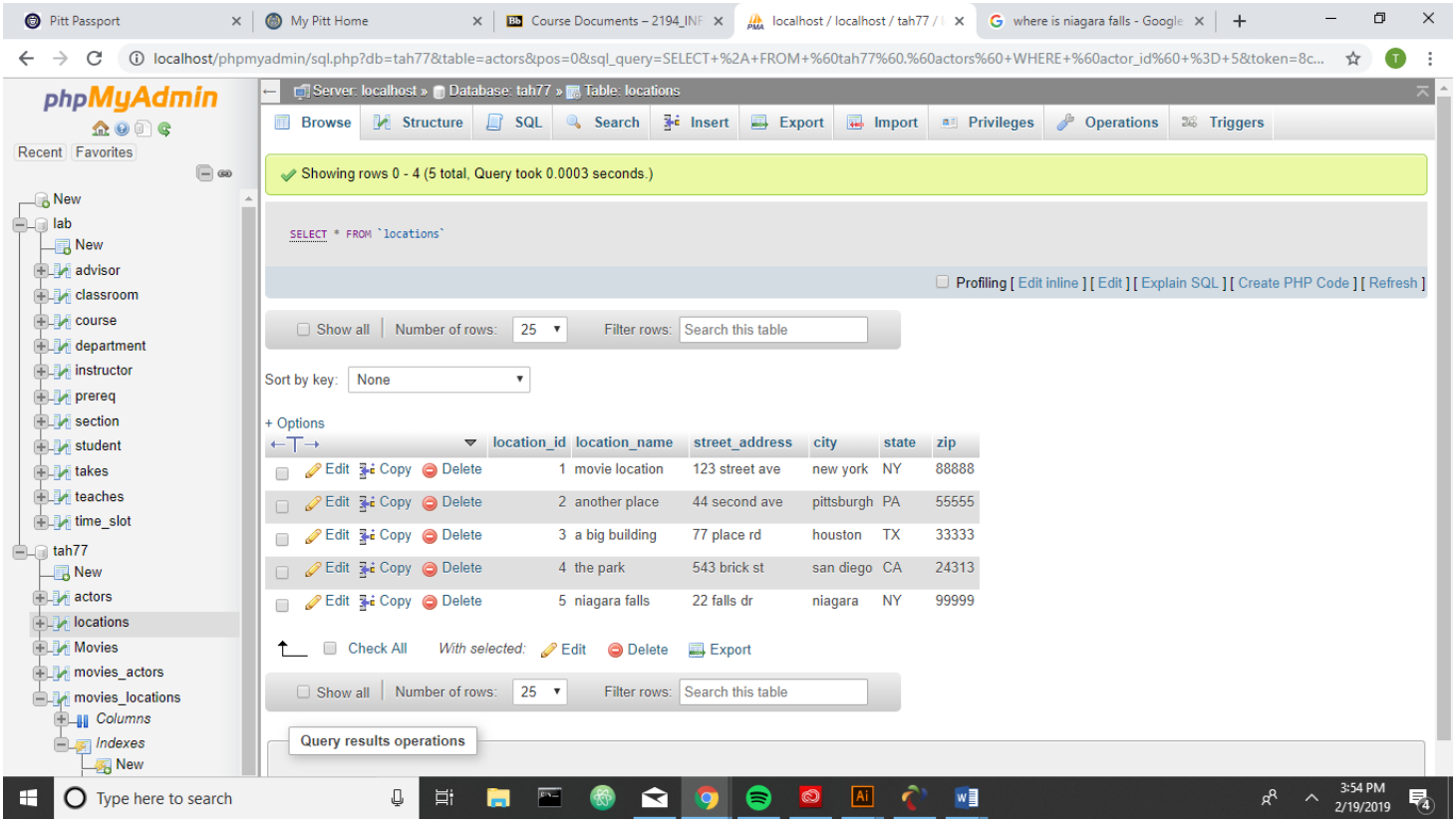
insert into actors values ('3', 'Freeman', 'Makai', '1997-08-30 02:00:00', 'another biography');
insert into actors values ('4', 'Franklin', 'Evelyn', '1988-10-12 03:00:00', 'evelyn franklyn biography');
insert into actors values ('5', 'Moro', 'Jed', '1990-08-11 01:00:00', 'jed moro biography');



2. At least 5 locations in the **locations** table

insert into locations values ('1', 'movie location', '123 street ave', 'new york', 'NY', '88888');
insert into locations values ('2', 'another place', '44 second ave', 'pittsburgh', 'PA', '55555');
insert into locations values ('3', 'a big building', '77 place rd', 'houston', 'TX', '33333');
insert into locations values ('4', 'the park', '543 brick st', 'san diego', 'CA', '24313');

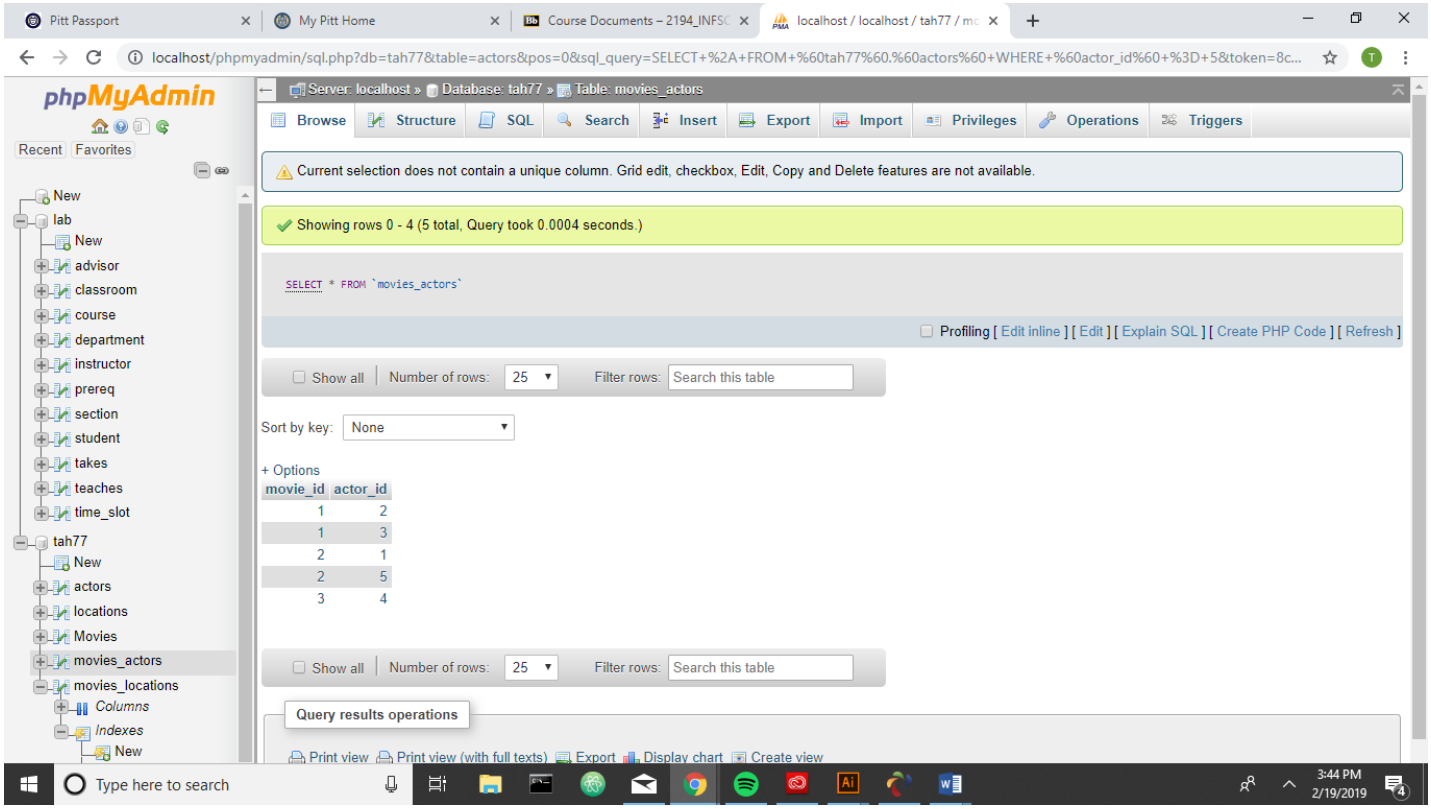
insert into locations values ('5', 'niagara falls', '22 falls dr', 'niagara', 'NY', '99999');



Task 4 (10 points): For each junction table, create at least five relationships (insert at least five rows of proper IDs).

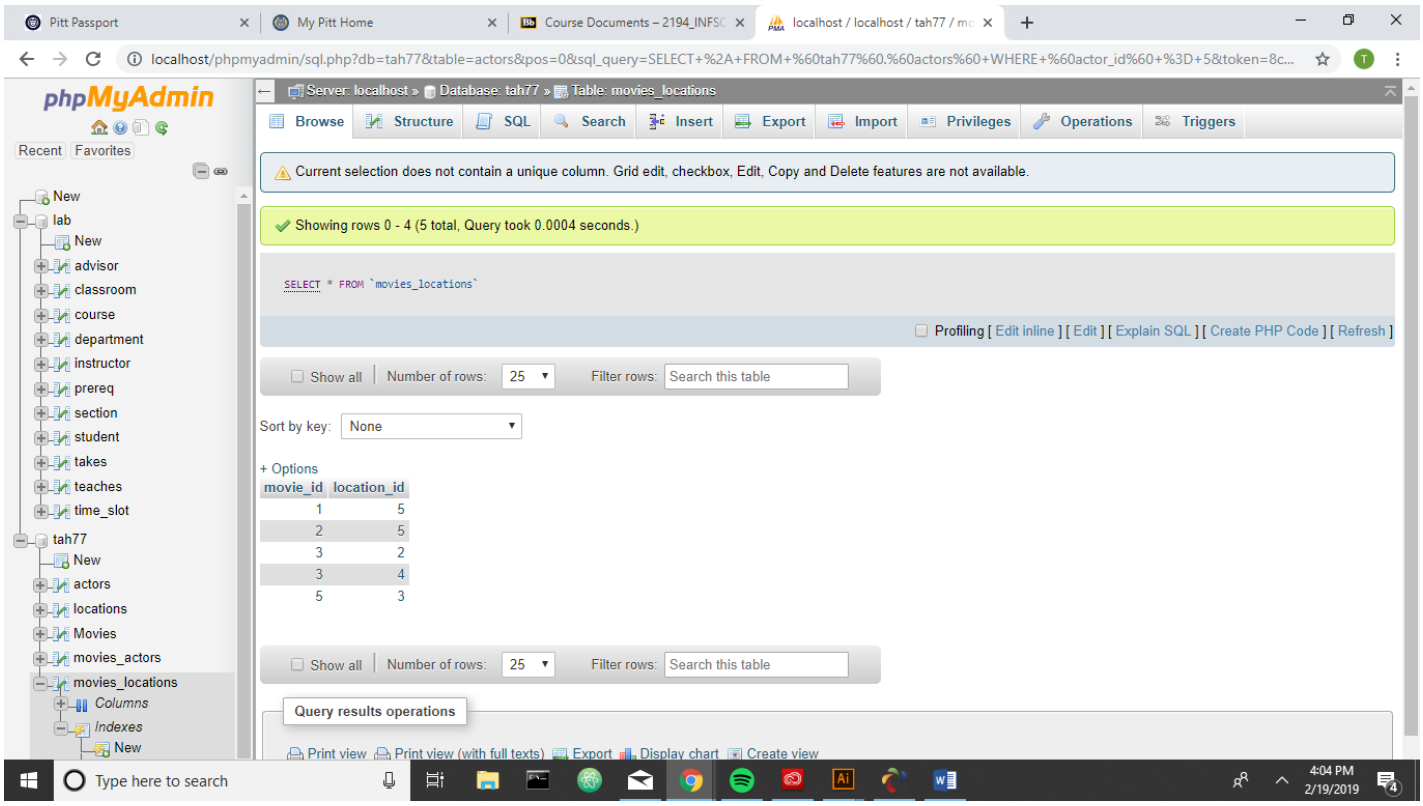
movies_actors

insert into movies_actors (movie_id, actor_id) values (1, 2)
insert into movies_actors (movie_id, actor_id) values (1, 3)
insert into movies_actors (movie_id, actor_id) values (2, 1)
insert into movies_actors (movie_id, actor_id) values (2, 5)
insert into movies_actors (movie_id, actor_id) values (3, 4)



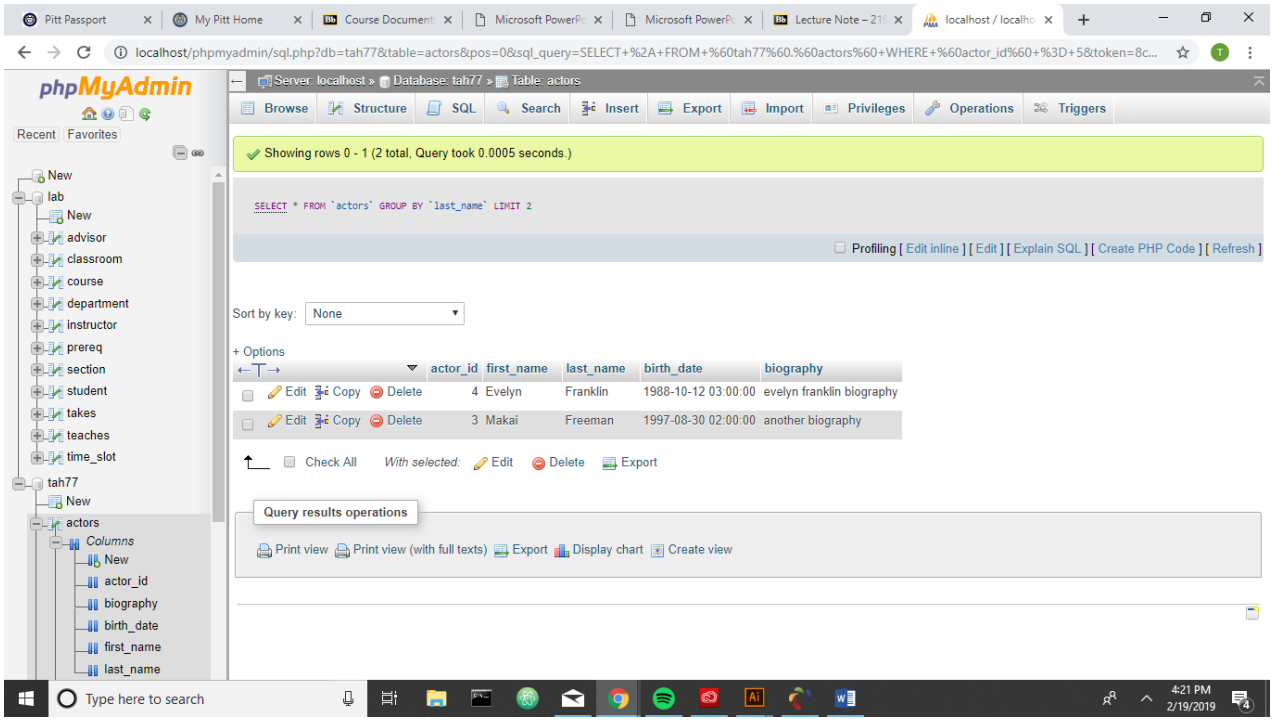
movies_locations:

insert into movies_locations (movie_id, location_id) values (1, 5);
insert into movies_locations (movie_id, location_id) values (2, 5);
insert into movies_locations (movie_id, location_id) values (3, 2);
insert into movies_locations (movie_id, location_id) values (3, 4);
insert into movies_locations (movie_id, location_id) values (5, 3);

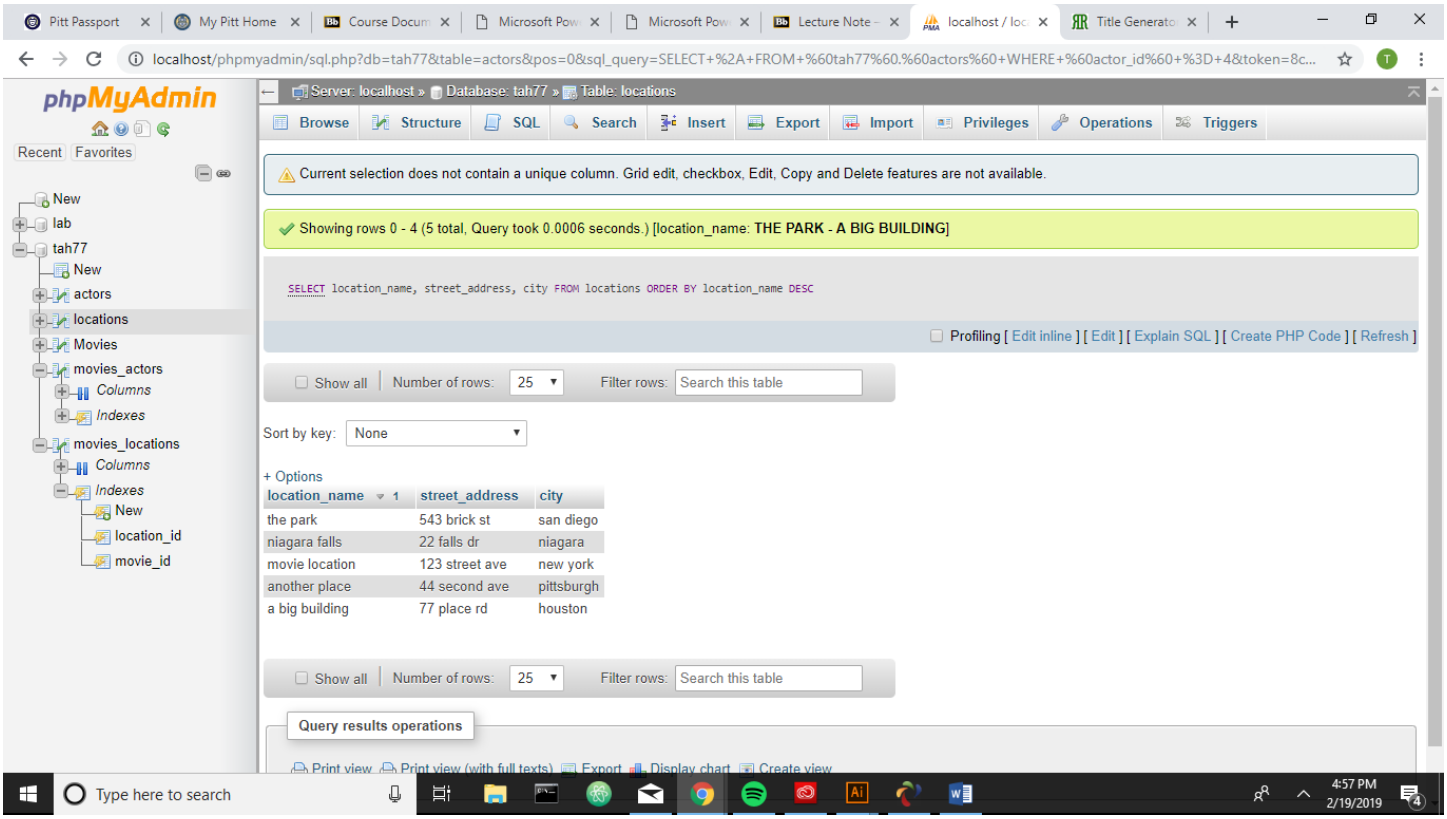


Task 5 (5 points): Write a SELECT statement to display top 2 actors sorted by actor's last name.

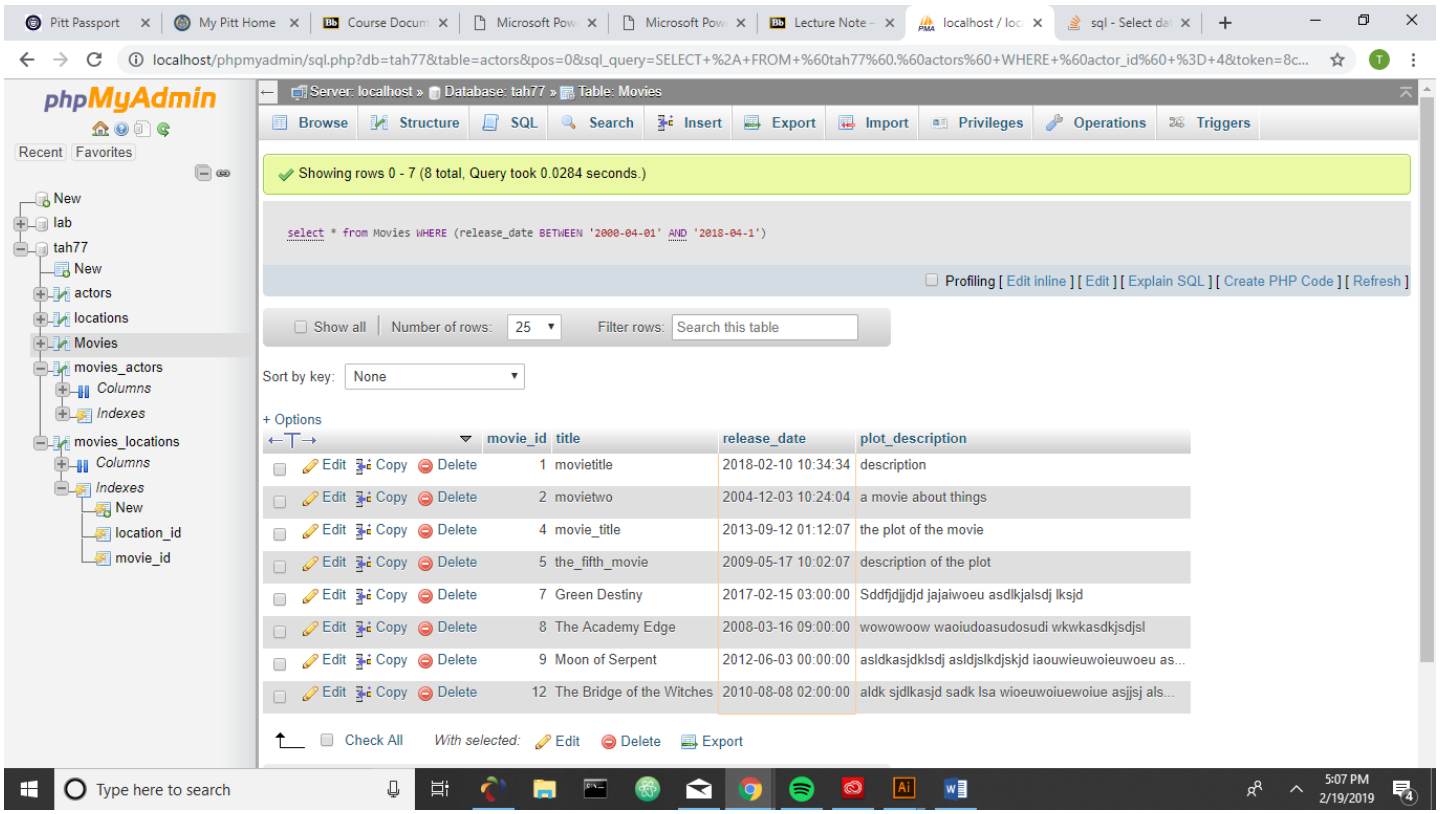
SELECT * FROM `actors` GROUP BY `last_name` LIMIT 2;



Task 6 (5 points): Write a SELECT statement to display location name, street address, and city sorted by location name in descending order.
SELECT location_name, street_address, city FROM locations ORDER BY location_name DESC

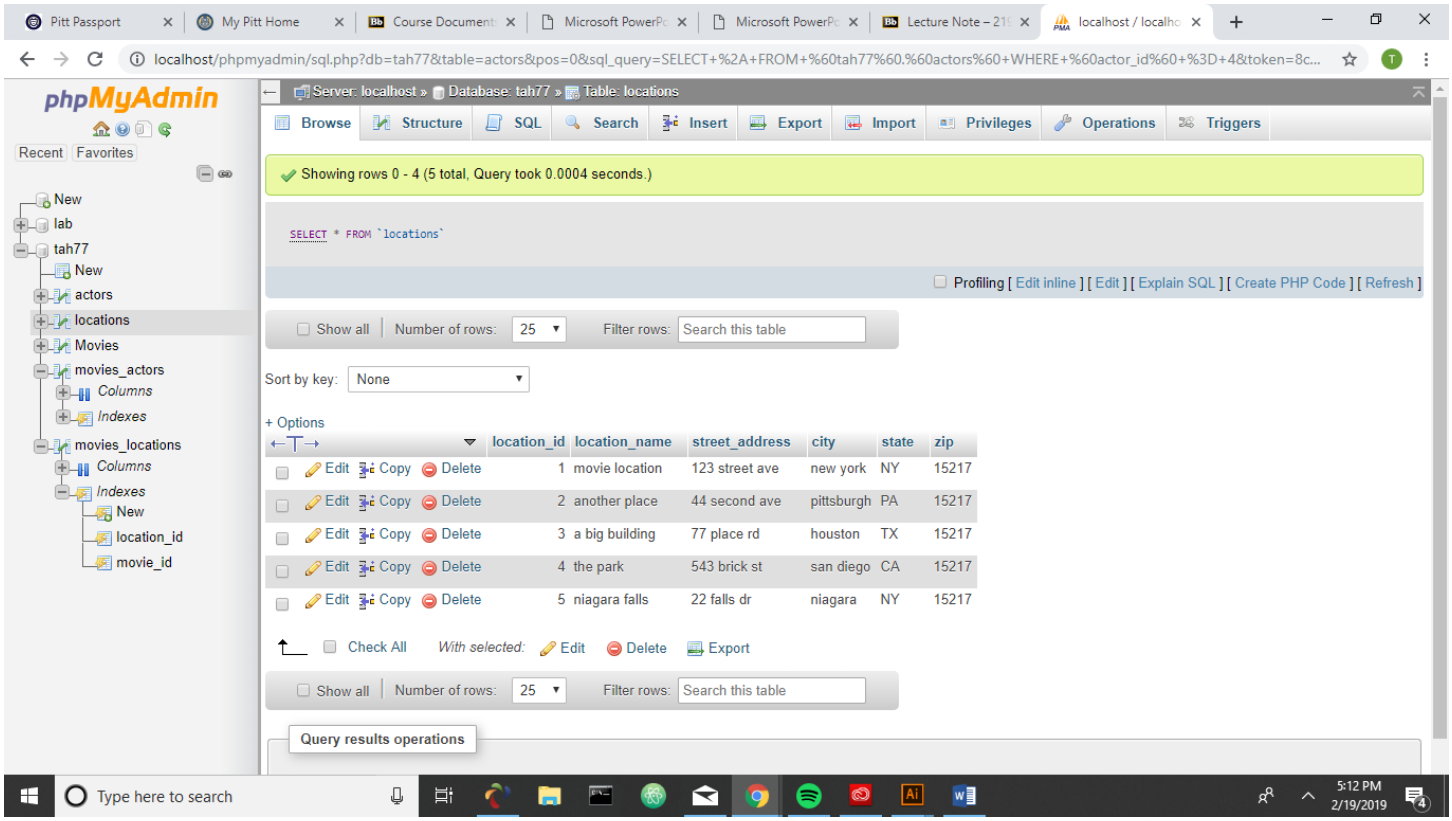


Task 7 (5 points): Write a SELECT statement to display movies released between two dates of your choice.
select * from Movies WHERE (release_date BETWEEN '2000-04-01' AND '2018-04-1')



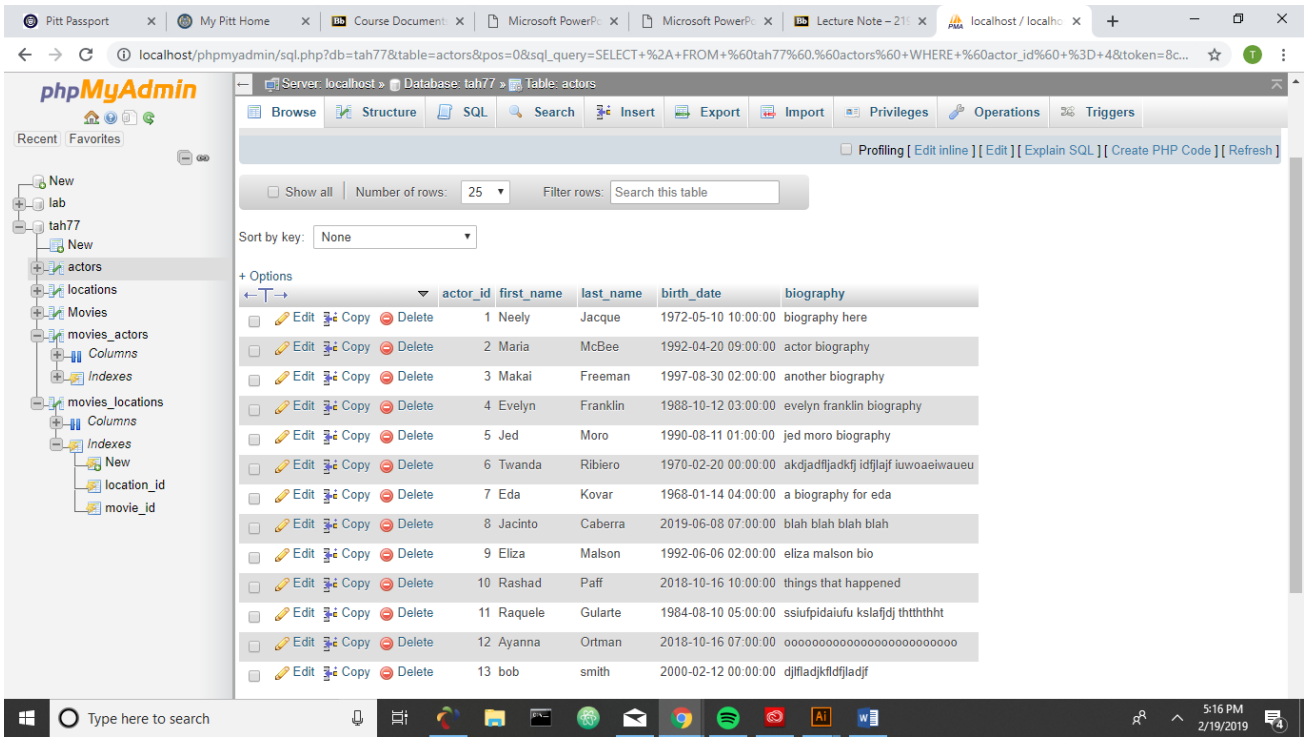
Task 8 (5 points): Write an UPDATE statement to update zip code for all locations to 15217

UPDATE locations SET zip='15217';

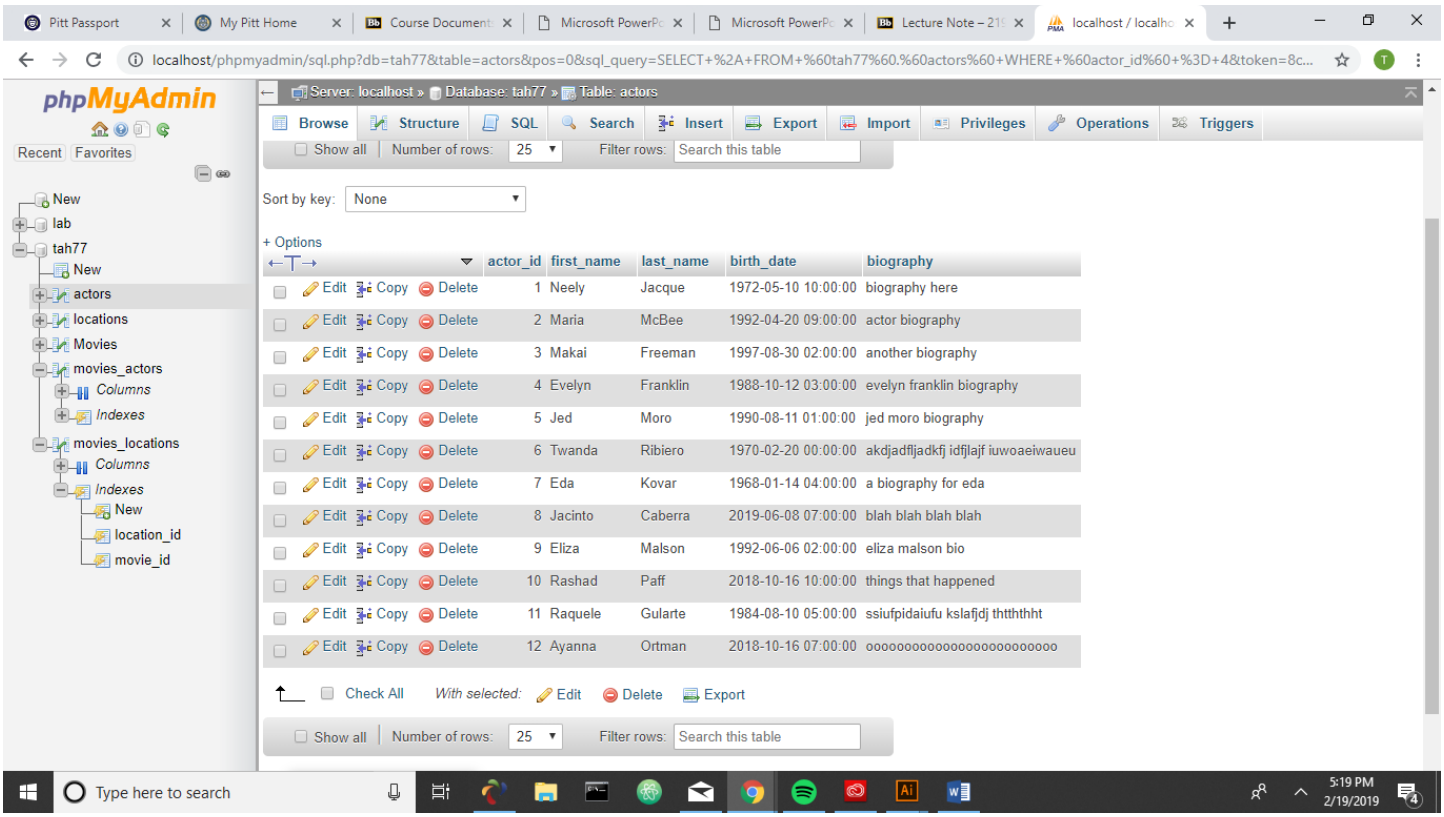


Task 9 (5 points): Write a statement that deletes one row of your choice from the actor's table. Be careful – make sure to write a correct WHERE clause.

DELETE FROM `actors` WHERE actor_id = '13';



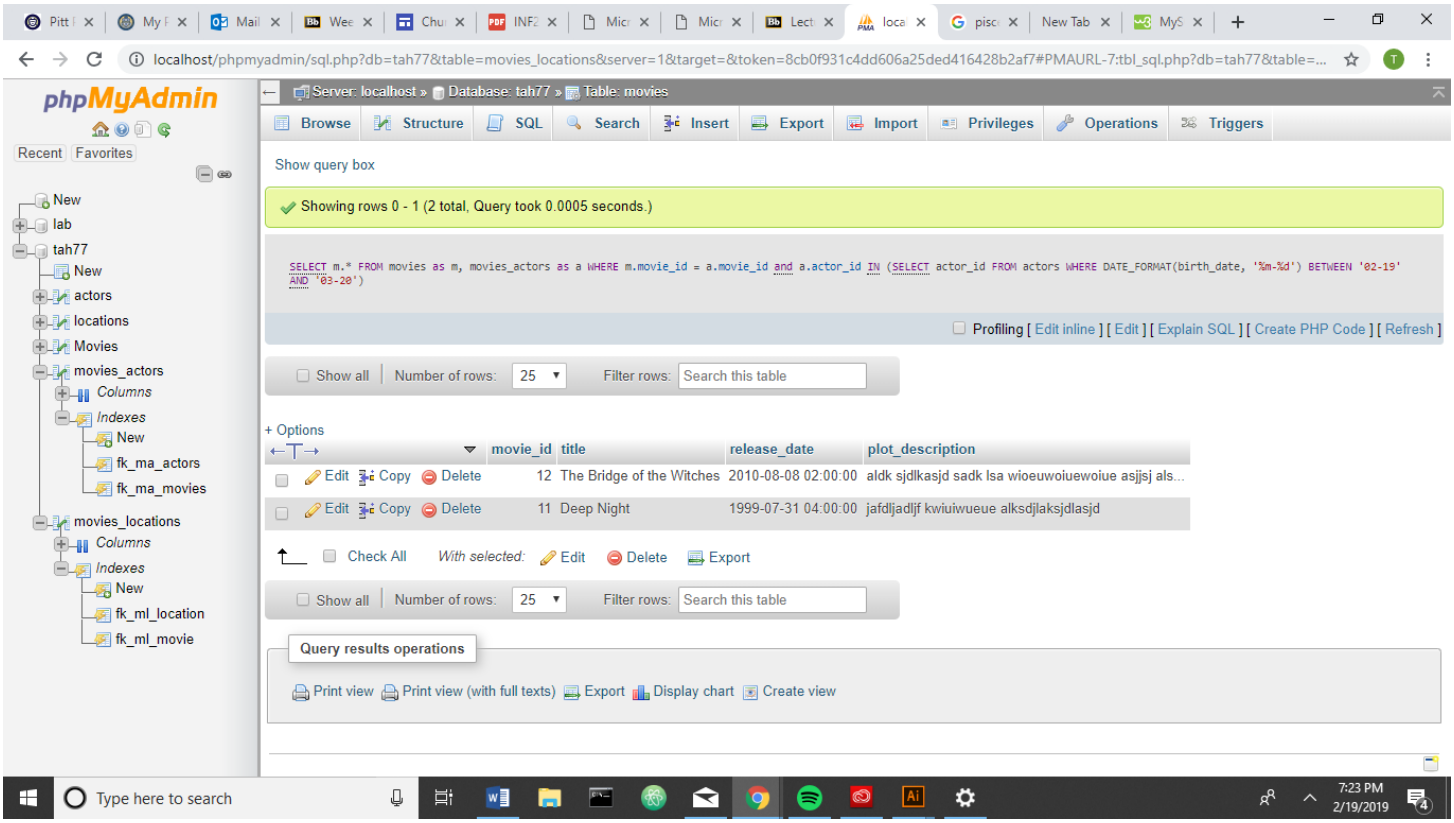
Before (actor_id = 13 exists in table)



Task 10 (5 Points): Write a SELECT statement to display movies which are played by any actor/actress who is a Gemini (or pick one Horoscope Sign you like)

***shows movies with pisces actors**

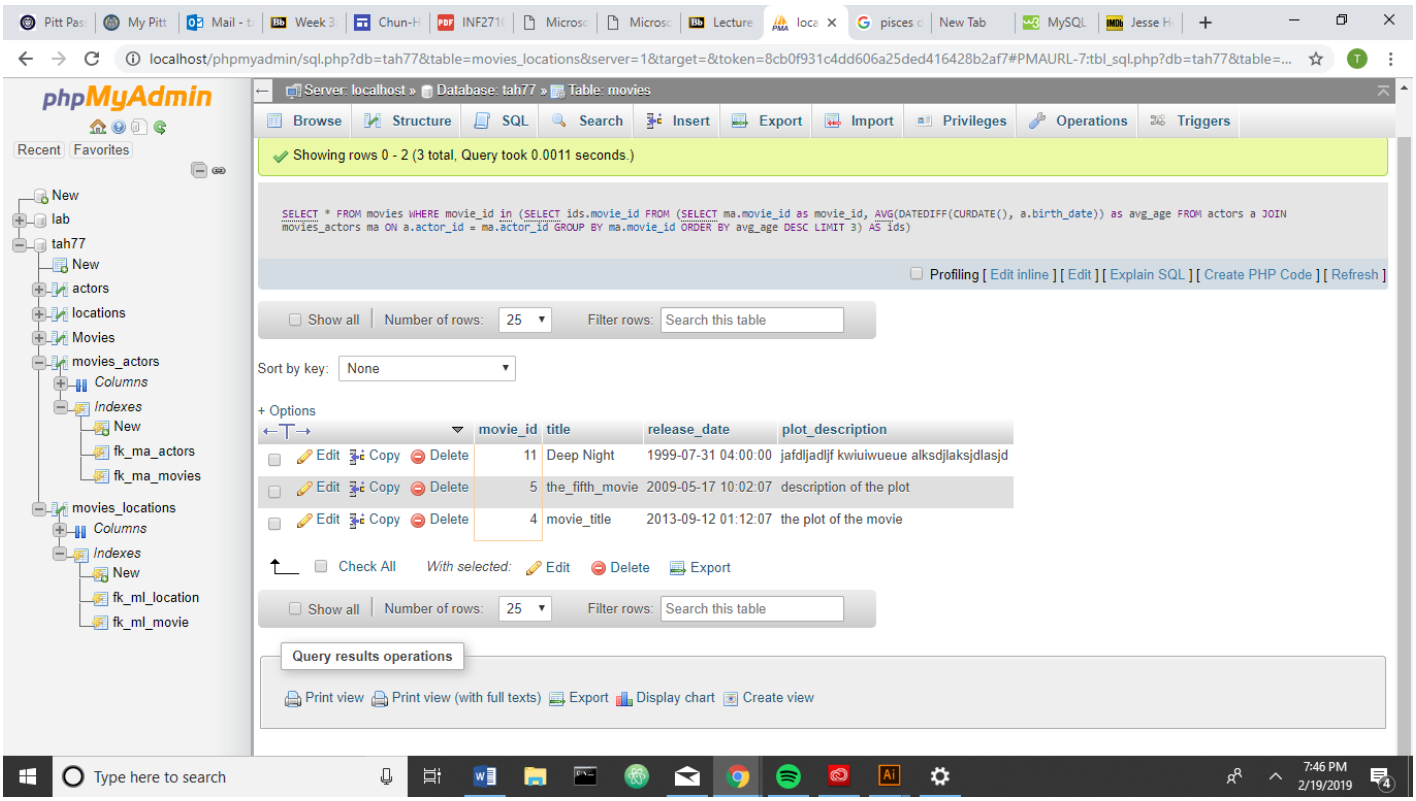
SELECT m.* FROM movies as m, movies_actors as a WHERE m.movie_id = a.movie_id and a.actor_id IN (SELECT actor_id FROM actors WHERE DATE_FORMAT(birth_date, '%m-%d') BETWEEN '02-19' AND '03-20')



Task 11 (5 points): Write a SELECT statement to display movies which the main cast (all actor/actress) has the highest average age.

***Shows top 3 movies with highest avg age**

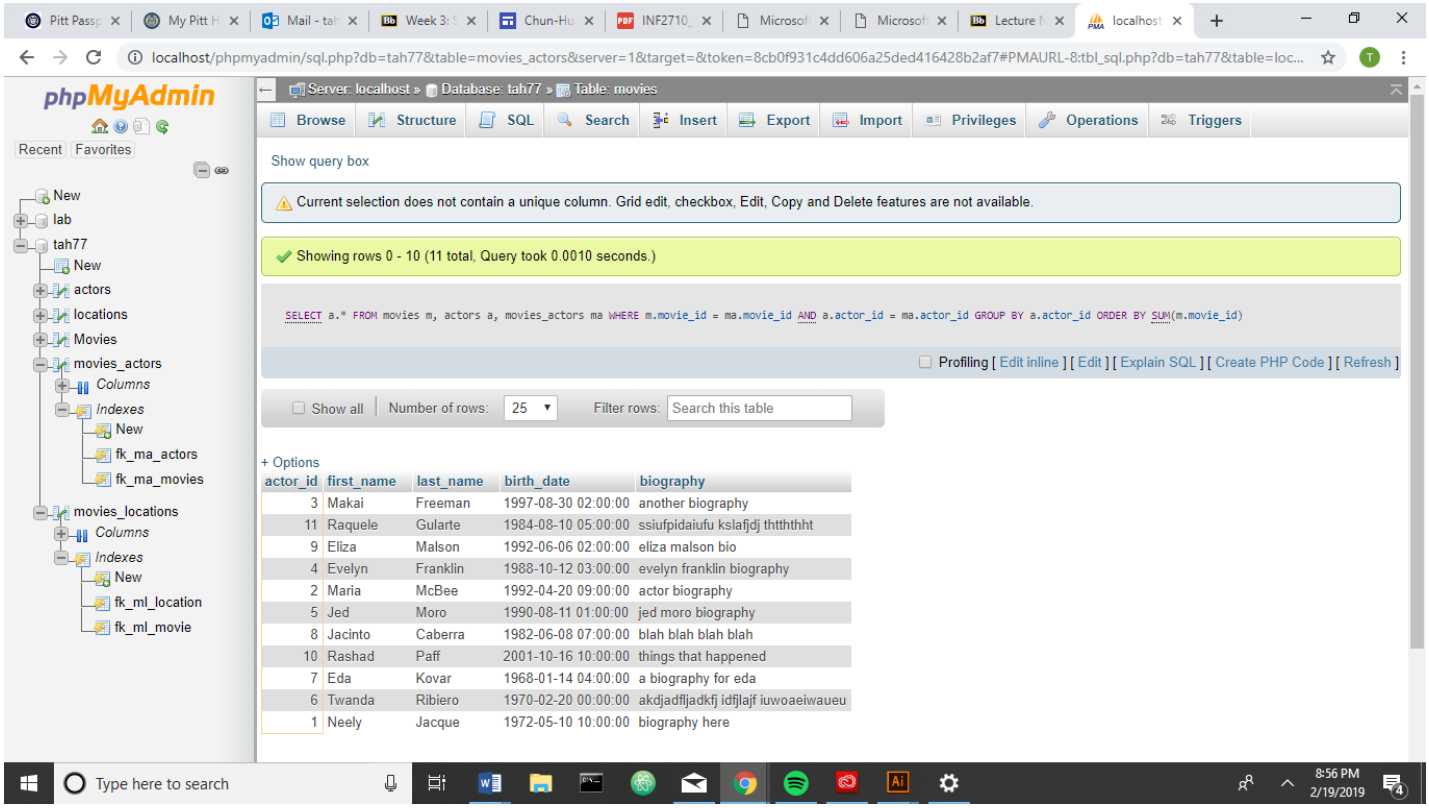
SELECT * FROM movies WHERE movie_id in (SELECT ids.movie_id FROM
(SELECT ma.movie_id as movie_id, AVG(DATEDIFF(CURDATE(), a.birth_date)) as avg_age
FROM actors a
JOIN movies_actors ma ON a.actor_id = ma.actor_id GROUP BY ma.movie_id ORDER BY avg_age DESC LIMIT 3) AS ids)



Task 12 (5 points): Write a SELECT statement to display the Greatest Extra (http://www.imdb.com/title/tt3087298/) in your movie database, i.e., the actors who have appeared in the most movies.

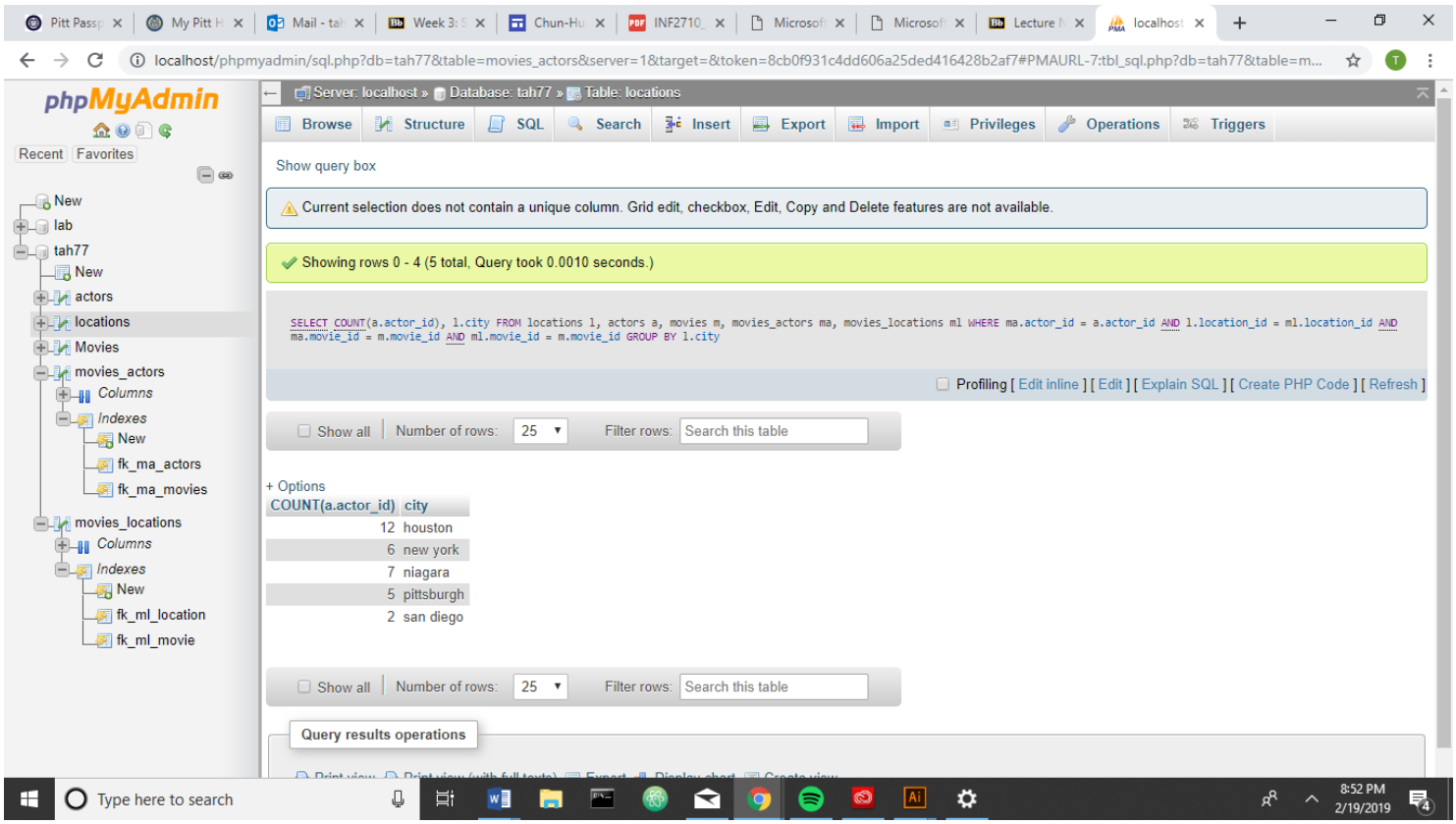
***shows list of actors in descending order of # of movies (actor_id = 3 is the 'greatest extra' currently)**

SELECT a.* FROM movies m, actors a, movies_actors ma WHERE m.movie_id = ma.movie_id AND a.actor_id = ma.actor_id GROUP BY a.actor_id ORDER BY SUM(m.movie_id);



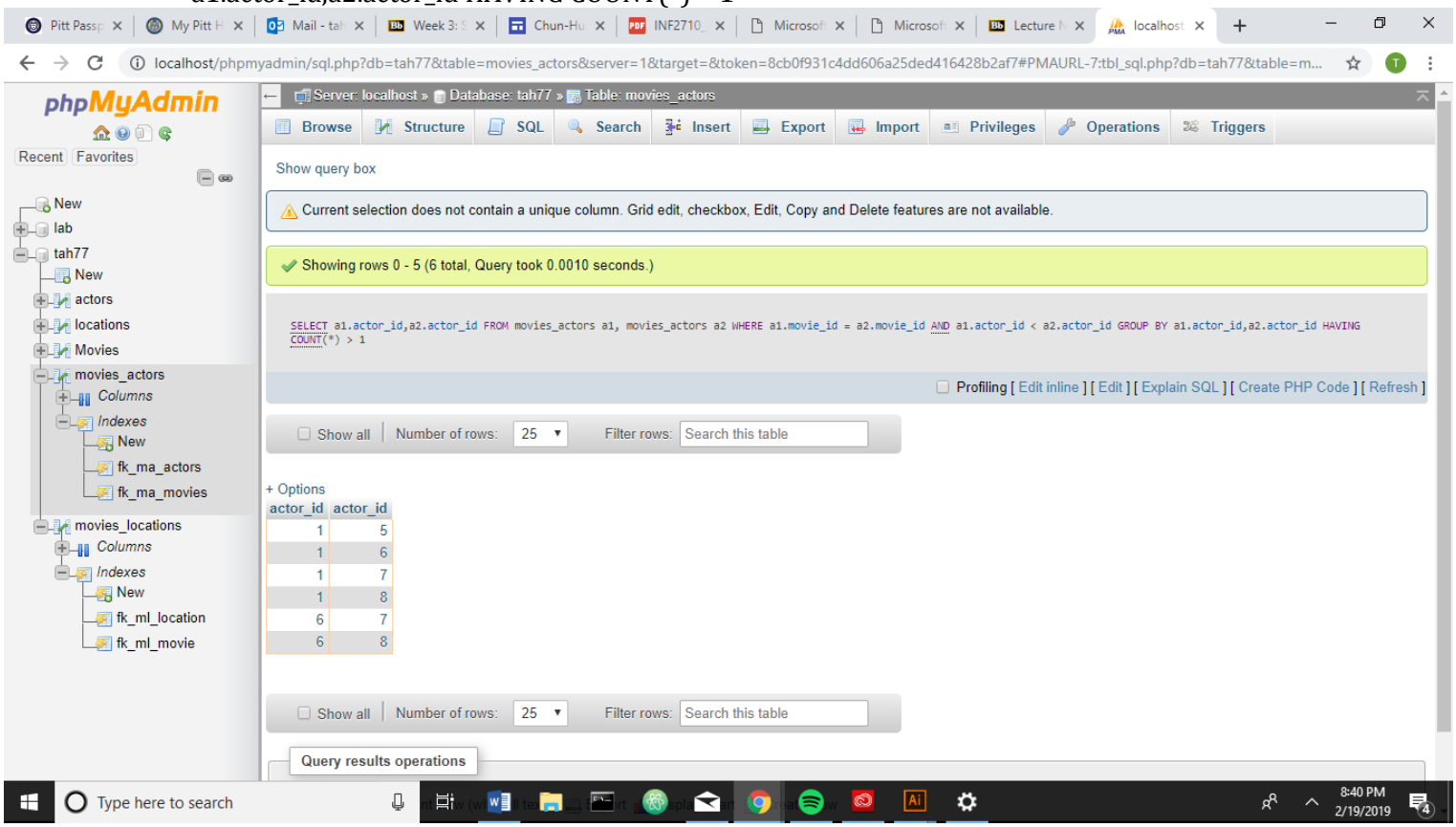
Task 13 (5 Points): Write a SELECT statement to display the number of actor occurrence count of each city, i.e., how many actors ever appear in each city? [Output: locations.city, "number of actor occurrence count"]

SELECT COUNT(a.actor_id), l.city FROM locations l, actors a, movies m, movies_actors ma, movies_locations ml WHERE ma.actor_id = a.actor_id AND l.location_id = ml.location_id AND ma.movie_id = m.movie_id AND ml.movie_id = m.movie_id GROUP BY l.city



Task 14 (5 Points): Write a SELECT statement to display an actor-to-actor network who ever act more than once in the same movie, i.e., to show a list of actor ID pair (ActorID1, ActorID2) that co-act more than one time.

SELECT a1.actor_id,a2.actor_id FROM movies_actors a1, movies_actors a2
WHERE a1.movie_id = a2.movie_id AND a1.actor_id < a2.actor_id GROUP BY
a1.actor_id,a2.actor_id HAVING COUNT(*) > 1



***Hint: for some questions, you may need to insert more data so that you can see the meaningful output.**