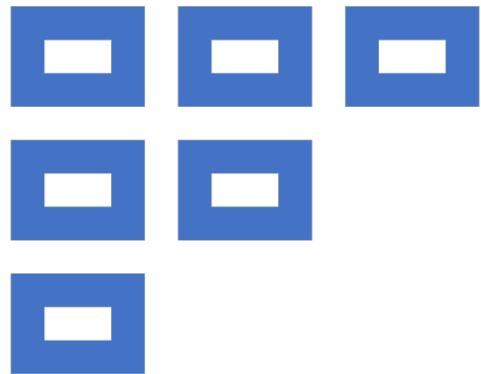

CAMPUS EATS PROJECT

INTRODUCTION TO DATABASE
SYSTEMS: DR. THOMPSON

TONY
SHANIQUA
PEARCE





THE FOCUS OF THE PROJECT IS
ON THE CORRECT DESIGN
AND DEVELOPMENT OF A
DATABASE USING MYSQL

CAMPUS
MEMBERS
ORDER FOOD
AND VIEW
RATINGS



**DRIVERS:
STUDENTS
APPROVED TO
DRIVE, VIEW
THEIR DRIVE
RATINGS**



[This Photo](#) by Unknown Author is licensed under [CC BY-NC](#)

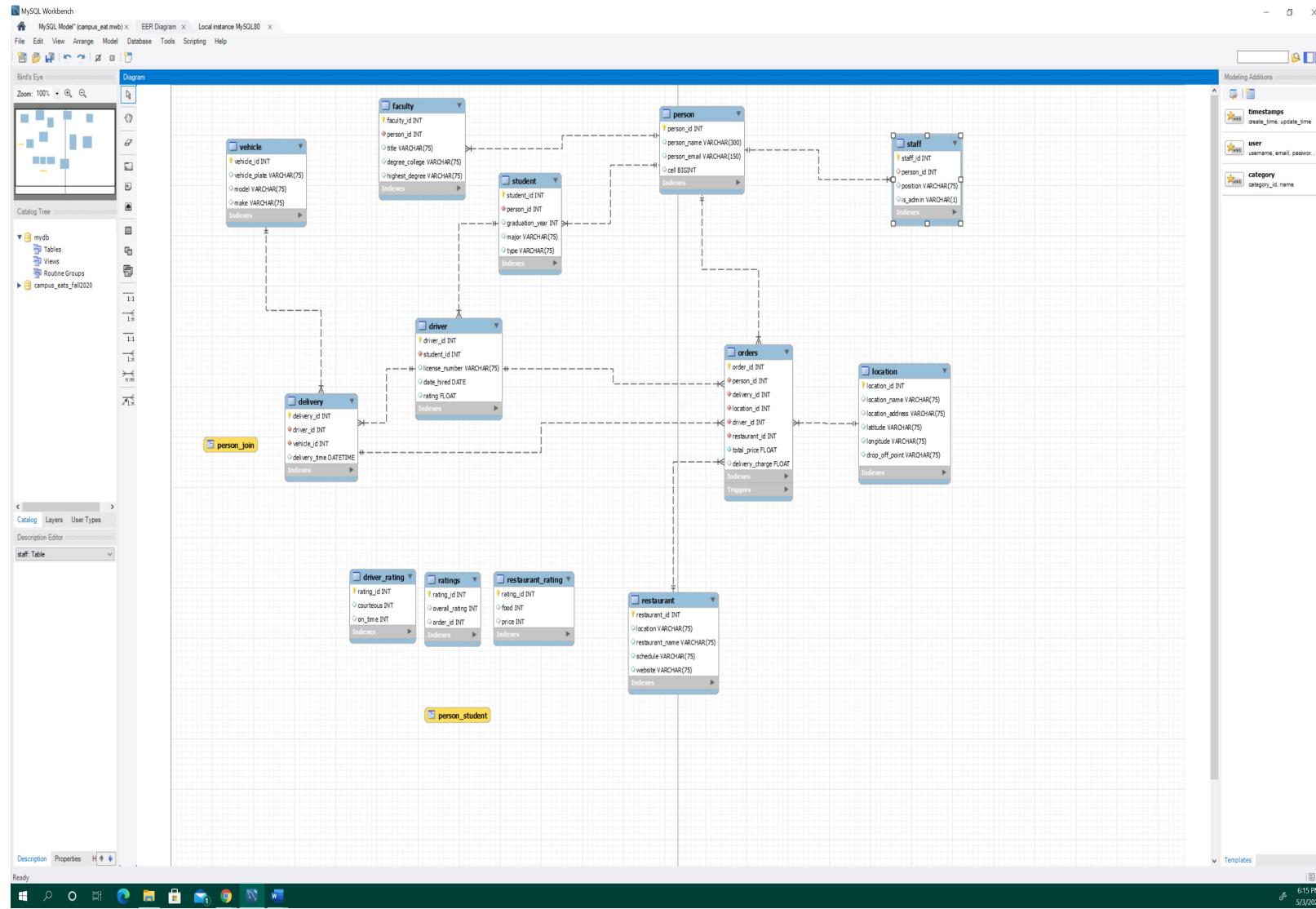
RESTAURANTS:
GET APPROVED
TO JOIN THE
SYSTEM



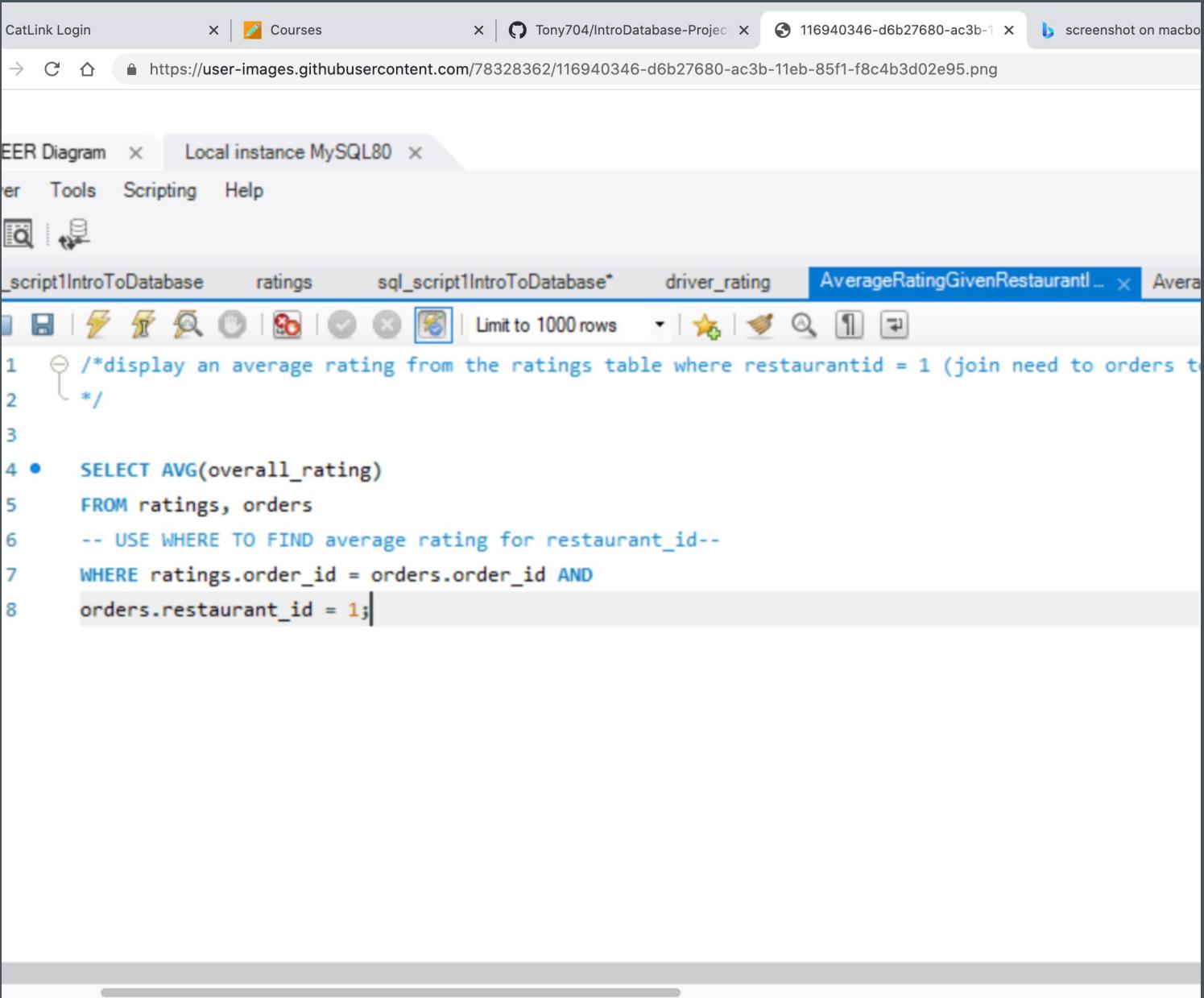
TOP 3 REASONS
why our clients
CHOOSE US

This Photo by Unknown Author is licensed under [CC BY-NC-ND](#)

HERE IS A PICTURE OF
THE ENHANCED
ENTITY DIAGRAM
THAT WE USED TO
CREATE THE
DATABASE



WE USED MY SQL TO CREATE QUERIES:



The screenshot shows the MySQL Workbench interface. The title bar includes tabs for "EER Diagram", "Local instance MySQL80", and "sql_script1IntroToDatabase*". The main area displays an SQL query:

```
1 /*display an average rating from the ratings table where restaurantid = 1 (join need to orders to get overall rating) */
2 *
3
4 • SELECT AVG(overall_rating)
5   FROM ratings, orders
6   -- USE WHERE TO FIND average rating for restaurant_id--
7   WHERE ratings.order_id = orders.order_id AND
8     orders.restaurant_id = 1;
```

The code uses a self-join between the 'ratings' and 'orders' tables to calculate the average rating for a specific restaurant (restaurant_id = 1). The query is annotated with comments explaining the purpose and the join condition.

QUERIES CONT.

The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query editor contains the following SQL code:

```
1  /* display an average rating from the restaurant table where restaurantid = 1 (join)
2  */
3
4 • SELECT AVG(overall_rating)
5   FROM ratings, orders, restaurant_rating
6   -- USE WHERE TO FIND average rating for from the restaurant table--
7   WHERE restaurant_rating.rating_id = orders.restaurant_id AND
8     orders.restaurant_id = 1;
9
```

The results grid below the query editor displays the output of the query:

AVG(overall_rating)
2.9500

QUERIES

The screenshot shows a MySQL Workbench interface with a connection to a Local instance MySQL 8.0. The query editor contains the following SQL code:

```
/* display all average rating from the drivers table where restaurantid = 1 (join needed . . . )
*/
SELECT AVG(rating)
FROM ratings, orders, driver
-- USE WHERE TO FIND average rating for from the drivers table--
WHERE driver.driver_id = orders.driver_id AND
orders.restaurant_id = 1;
```

QUERIES

The screenshot shows a browser window with several tabs open. The active tab is titled "sql_script1IntroToDatabase" and contains a SQL script. The script is a CREATE VIEW statement named "Orders_order". It selects "order_id", "p.person_id", "person_name", "r.restaurant_id", and "r.restaurant_name" from the "orders" table, joining it with the "person" and "restaurant" tables based on their respective IDs. The results are ordered by "person_name" in ascending order.

```
1  /*display all of the orders made by a customer over a week
2  */
3  • CREATE VIEW Orders_order
4  AS SELECT order_id, p.person_id, person_name, r.restaurant_id, r.restaurant_name
5  FROM orders
6  INNER JOIN person AS p
7      ON orders.person_id = p.person_id
8  INNER JOIN restaurant AS r
9      ON orders.restaurant_id = r.restaurant_id
10 ORDER BY person_name ASC
```

QUERIES

```
1  /* An original query of your choice including at least 5 tables
2   Get the order_id, faculty name, restaurant name, location address
3   restaurant.restaurant_name, location.location_address for the faculty member from their first
4   */
5
6 • SELECT order_id, person_name, faculty.title, restaurant.restaurant_name, restaurant.location, delivery.delivery_time
7   FROM orders
8   INNER JOIN person -- USE Inner join to match order_id with person_id
9       ON orders.order_id = person.person_id
10
11  RIGHT JOIN faculty -- RIGHT JOIN to find the faculty name
12      ON orders.person_id = faculty.person_id
13
14  LEFT JOIN restaurant -- LEFT JOIN to get the restaurant name and location
15      ON orders.restaurant_id = restaurant.restaurant_id
16
17  LEFT JOIN delivery -- LEFT JOIN to get the delivery time
18      ON orders.driver_id = delivery.driver_id
19
20 ORDER BY person_name ASC, delivery_time ASC
```