

## Team Member

Parth Patel

Het Thakkar

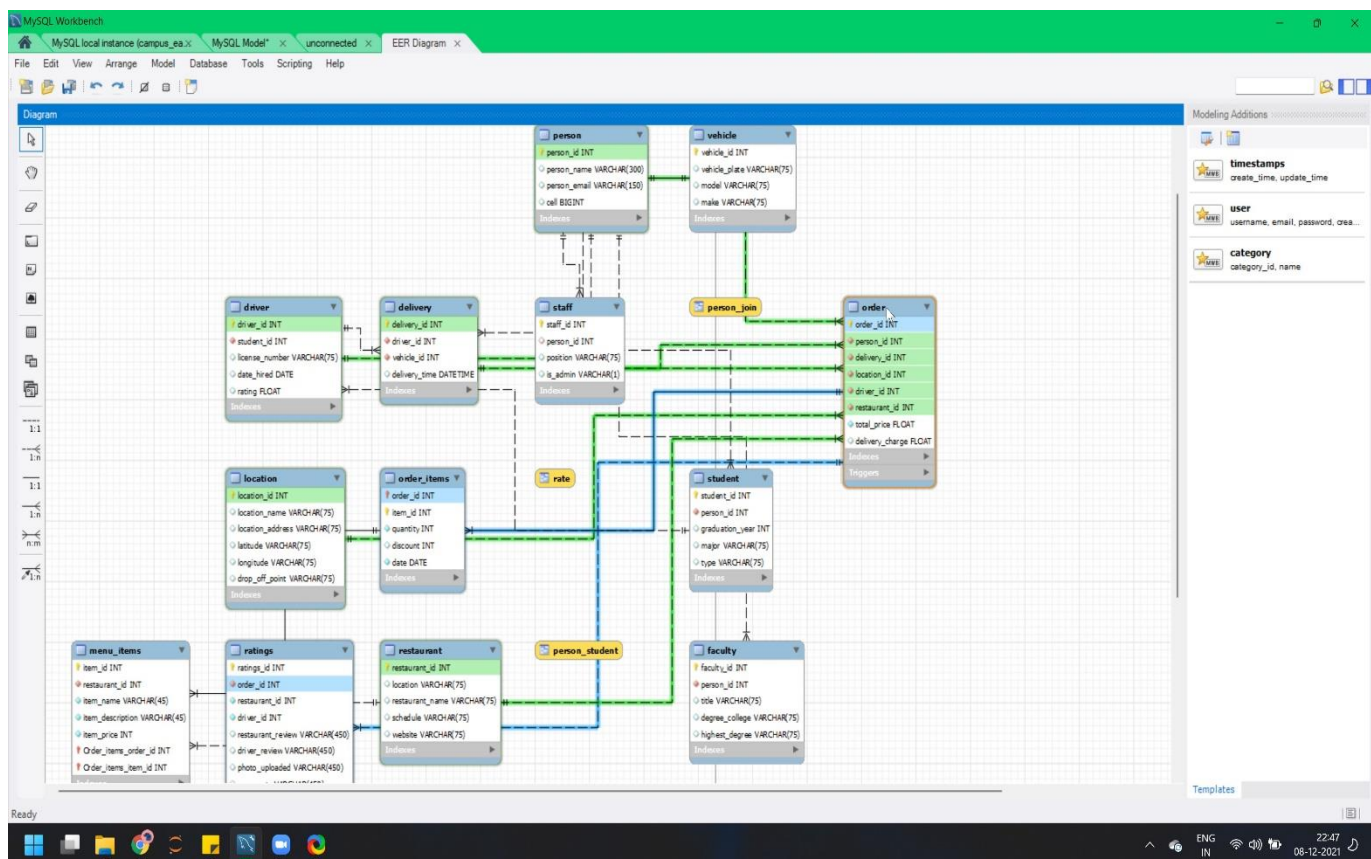
Tharani Kumaresan

Rishab Semrani

## Project Introduction:

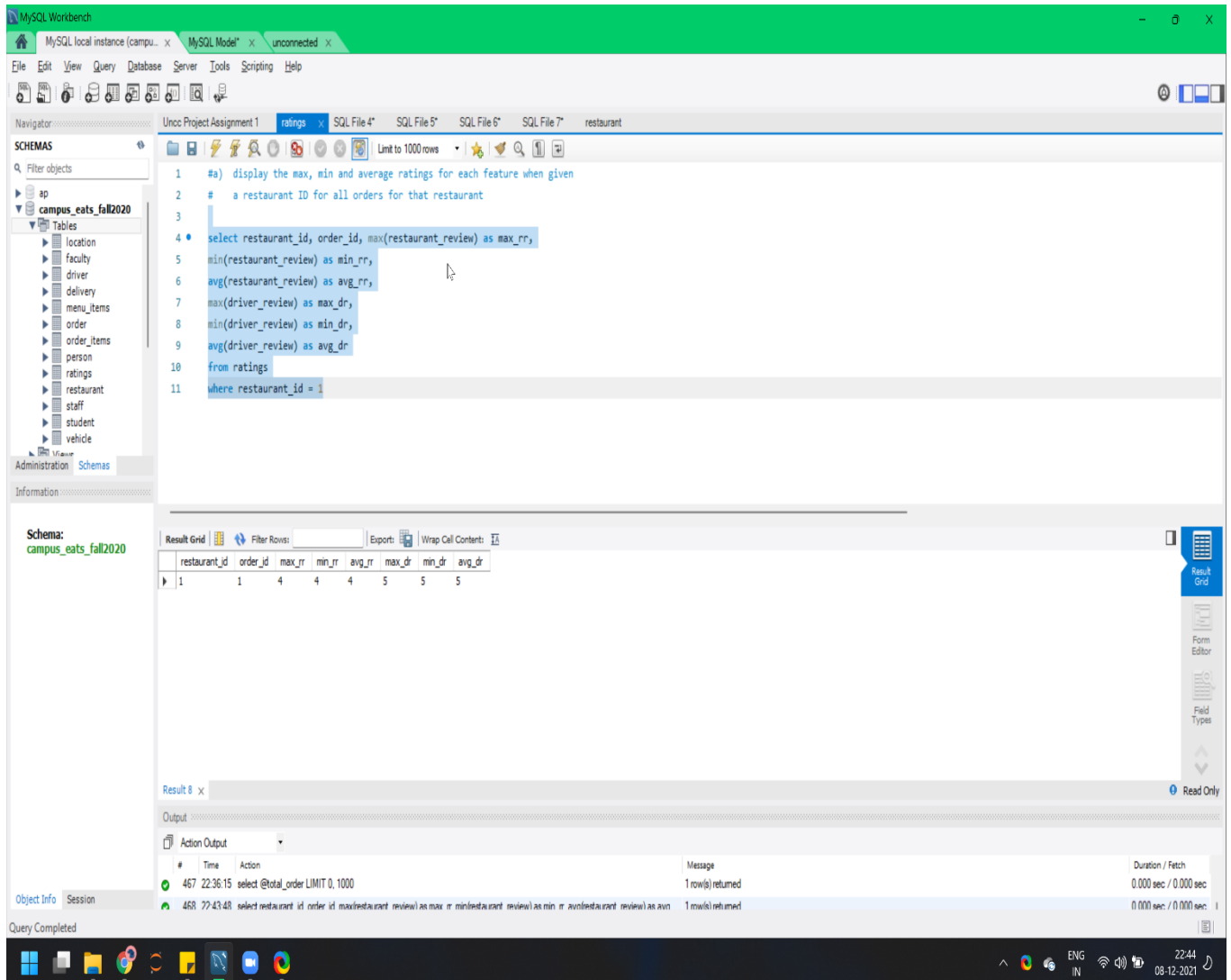
The goal of our projects is to create that manages campus controlled food delivery Service using SQL. we will be enhancing the current database with a rating system for both restaurants and delivery drivers. The database includes for person (like Student, Faculty, Driver), different locations, different restaurants and vehicles, delivery. MySQL Workbench is utilized to create this environment. MySQL was used for a majority of the SQL code that allowed us to have a visual representation of the tables, schemas and databases that we created.

## EERD:



## Stored Procedure:

[1] display the max, min and average ratings for each feature when given a restaurant ID for all orders for that restaurant



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'campus\_eats\_fall2020' database, including tables like 'location', 'faculty', 'driver', 'delivery', 'menu\_items', 'order', 'order\_items', 'person', 'ratings', 'restaurant', 'staff', 'student', and 'vehicle'. The main editor window contains a SQL query:

```
1 #a) display the max, min and average ratings for each feature when given
2 # a restaurant ID for all orders for that restaurant
3
4 select restaurant_id, order_id, max(restaurant_review) as max_rr,
5 min(restaurant_review) as min_rr,
6 avg(restaurant_review) as avg_rr,
7 max(driver_review) as max_dr,
8 min(driver_review) as min_dr,
9 avg(driver_review) as avg_dr
10 from ratings
11 where restaurant_id = 1
```

The 'Result Grid' at the bottom shows the output of the query:

restaurant_id	order_id	max_rr	min_rr	avg_rr	max_dr	min_dr	avg_dr
1	1	4	4	4	5	5	5

The 'Output' panel at the bottom shows the execution log with two entries:

- 467 22:36:15 select @total\_order LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
- 468 22:43:48 select restaurant\_id order\_id max(restaurant\_review) as max\_rr min(restaurant\_review) as min\_rr avg(restaurant\_review) as avg\_rr 1 row(s) returned 0.000 sec / 0.000 sec

The status bar at the bottom indicates 'Query Completed'.

[2] display a count of the orders made by a customer for a specified date range when given a customer id

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'campus\_eats\_fall2020' database, including tables like location, faculty, driver, delivery, menu\_items, order, order\_items, person, ratings, restaurant, staff, student, and vehicle. The main editor window contains a SQL query:

```
1 -- b) display a count of the orders made by a customer for a specified date
2 -- range when given a customer id
3
4 SELECT count(o.person_id) FROM campus_eats_fall2020.order o
5 join order_items oi
6 on o.order_id = oi.order_id
7 where o.person_id = 1
8 and date between '2020-10-22' and '2021-10-22'
9 group by o.person_id;
```

The 'Result Grid' at the bottom shows the query results:

count(o.person_id)
2

The 'Output' panel at the bottom shows the execution log with two entries:

#	Time	Action	Message	Duration / Fetch
468	22:43:48	select restaurant_id, order_id, max(restaurant_review) as max_r, min(restaurant_review) as min_r, avg(restaurant_review) as avg...	1 row(s) returned	0.000 sec / 0.000 sec
469	22:44:13	SP1 FCT count(n person_id) FROM campus_eats_fall2020 order o join order_items oi on o.order_id = oi.order_id where o.person...	1 row(s) returned	0.000 sec / 0.000 sec

The status bar at the bottom indicates 'Query Completed'.

[3] display total price of the orders by each customer (distinct) for a specified date range

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'campus\_eats\_fall2020' database, including tables like 'location', 'faculty', 'driver', 'delivery', 'menu\_items', 'order', 'order\_items', 'person', 'ratings', 'restaurant', 'staff', 'student', and 'vehicle'. The main editor window contains a SQL query:

```
1 -- c) display total price of the orders by each customer (distinct) for a
2 -- specified date range
3
4 SELECT sum(o.total_price) FROM campus_eats_fall2020.order o
5 Join order_items oi
6 on o.order_id = oi.order_id
7 where date between '2020-10-22' and '2021-10-22'
8 group by o.person_id;
```

The 'Result Grid' at the bottom shows the query results:

sum(o.total_price)
21.630000114440918
18.030000686645508

The 'Output' panel at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
469	22:44:13	SELECT count(p.person_id) FROM campus_eats_fall2020.order o Join order_items oi on o.order_id = oi.order_id where o perso...	1 row(s) returned	0.000 sec / 0.000 sec
470	22:44:35	SFI FCT numto total price) FROM campus_eats_fall2020.order o Join order_items oi on o.order_id = oi.order_id where date bet	2 row(s) returned	0.000 sec / 0.000 sec

The status bar at the bottom indicates 'Query Completed'.

[4] display a particular customer's rating for a restaurant

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'campus\_eats\_fall2020' selected. The main editor window contains a SQL query (SQL File 6) that selects the person name, restaurant name, and restaurant review for a specific customer (person\_id=1) and restaurant (restaurant\_id=1). The query is as follows:

```
1  -- Execute the selected portion of the script or everything, if there is no selection
2
3  • Select p.person_name,res.restaurant_name,rat.restaurant_review
4  from campus_eats_fall2020.restaurant res
5  Join campus_eats_fall2020.ratings rat
6  on res.restaurant_id = rat.restaurant_id
7  Join campus_eats_fall2020.order o
8  on rat.order_id = o.order_id
9  Join campus_eats_fall2020.person p
10 on p.person_id = o.person_id
11 where p.person_id=1
12 and res.restaurant_id=1
13 group by o.order_id;
```

The 'Result Grid' shows the following data:

person_name	restaurant_name	restaurant_review
Keith Turner	Rath Ltd	4

The 'Output' pane at the bottom shows the execution log, indicating that the query was completed successfully and returned 1 row(s).

## [5] View

View Name : rate

Code :

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'campus\_eats\_fall2020' schema selected. The main editor window shows the following SQL code:

```
1 -- e) Have one of the above requirements represented in a View--Query1
2
3 CREATE VIEW rate AS
4 select restaurant_id, order_id, max(restaurant_review) as max_rr,
5 min(restaurant_review) as min_rr,
6 avg(restaurant_review) as avg_rr,
7 max(driver_review) as max_dr,
8 min(driver_review) as min_dr,
9 avg(driver_review) as avg_dr
10 from ratings
11 where restaurant_id = 1;
```

The bottom output window shows the execution results:

#	Time	Action	Message	Duration / Fetch
470	22:44:35	SELECT sum(b total_price) FROM campus_eats_fall2020 order o Join order_items oi on o.order_id = oi.order_id where date bet...	2 row(s) returned	0.000 sec / 0.000 sec
471	22:44:44	Select n person_name as restaurant_name, restaurant_review from campus_eats_fall2020 restaurant res Join campus_eats_f	1 row(s) returned	0.000 sec / 0.000 sec

## [6] Stored Procedure

Stored Procedure Name: total\_order\_by\_customer

Code :

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'campus\_eats\_fall2020' selected. The main editor window shows the following SQL code:

```
1  -- stored procedure- total orders by customer
2
3  DROP PROCEDURE IF EXISTS total_order_by_customer
4  DELIMITER //
5  CREATE PROCEDURE total_order_by_customer ( in cus_id int, in datea VARCHAR(100), in dateb
6  VARCHAR(100), out total_order int)
7  BEGIN
8  SELECT COUNT(*)
9  INTO total_order
10 FROM campus_eats_fall2020.order o
11 JOIN order_items oi
12 ON o.order_id = oi.order_id
13 WHERE o.person_id = cus_id
14 AND date BETWEEN datea AND dateb;
15 END //
16 DELIMITER ;
17 set @cus_id=1, @datea='2020-10-22', @dateb='2021-10-22';
18 CALL total_order_by_customer(@cus_id, @datea, @dateb, @total_order);
19 select @total_order;
```

The bottom panel shows the 'Output' tab with the following results:

#	Time	Action	Message	Duration / Fetch
470	22:44:35	SELECT sum(o.total_price) FROM campus_eats_fall2020.order o JOIN order_items oi on o.order_id = oi.order_id where date bet...	2 row(s) returned	0.000 sec / 0.000 sec
471	22:44:44	Select n person: name res restaurant: name res restaurant: review: from campus_eats_fall2020.restaurant res JOIN campus_eats f...	1 row(s) returned	0.000 sec / 0.000 sec

Query Completed