

Project Report: List and Description

Team: Deeksha Umesh, Rama Suchitha Challa, Pavan Vooturi, Priyatham Kumar Gajula

CampusEats is a food delivery app built by team 17, we have extended the project by adding a rating system for the restaurants and menu to restaurants. We added 3 new tables to the existing 10 tables. The tables are as follows: delivery, driver, faculty, location, order, person, restaurant, staff, student, and vehicle. We added a rating, menu_item, order_menu table. menu table is an associative table that sits between menu and order table because of the many to many relationship. We have extended the project further by updating the eerd, adding advanced SQL statements such as stored procedures, views, and functions which are located in the queries folder in Git. We have also added a data dictionary which gives a description of the columns in the database.

Stored Procedure:

Name: get_delivery_details

Purpose: To get delivery details.

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel lists various database objects including 'driver', 'faculty', 'location', 'menu_item', 'order', 'orders_menu', 'person', 'rating', and 'rest_menu'. The 'orders_menu' table is selected, and its columns are listed: 'order_id' (int), 'restaurant_id' (int), 'person_id' (int), and 'menu_id' (int). The main panel displays SQL code for creating a stored procedure named 'get_delivery_details'. The code includes a 'USE' statement for 'campus_eats_fall2020', a 'DROP PROCEDURE IF EXISTS' statement, a 'DELIMITER \$\$' statement, a 'CREATE DEFINER' statement, a 'BEGIN' block, a 'SELECT' statement with joins, an 'INNER JOIN' clause, and an 'END\$\$' statement. The procedure is then called with 'call get_delivery_details(1);'.

```
1 • USE campus_eats_fall2020;
2
3 • Drop Procedure IF EXISTS get_delivery_details;
4
5 DELIMITER $$
6 • CREATE DEFINER='root'@'localhost' PROCEDURE `get_delivery_details`(in delivery_id varchar(100))
7 BEGIN
8 Select d.delivery_id, d.delivery_time, p1.person_name as driver, v.vehicle_plate, v.model
9 FROM campus_eats_fall2020.delivery AS d
10 INNER JOIN
11 person AS p1
12 ON d.driver_id = p1.person_id
13 AND d.delivery_id = delivery_id
14 INNER JOIN
15 vehicle as v
16 ON v.vehicle_id = d.vehicle_id
17 AND d.delivery_id = delivery_id;
18 END$$
19
20 • call get_delivery_details(1);
```

Result:

The screenshot shows a 'Result Grid' window displaying the output of the stored procedure. The grid has five columns: 'delivery_id', 'delivery_time', 'driver', 'vehicle_plate', and 'model'. The first row of data shows a delivery with ID 1, time 2004-04-15 15:15:01, driver Miss Rosanna Connelly, vehicle plate 9687, and model s.

delivery_id	delivery_time	driver	vehicle_plate	model
1	2004-04-15 15:15:01	Miss Rosanna Connelly	9687	s

Name: get_order_details

Purpose: To get order details.

The screenshot shows a database IDE with a left sidebar containing a schema tree. The 'orders_menu' table is selected, and its columns are listed: order_id (int), restaurant_id (int), person_id (int), and menu_id (int). The main editor displays the following SQL code:

```
1 • USE campus_eats_fall12020;
2
3 • Drop Procedure IF EXISTS get_order_details;
4
5 DELIMITER $$
6 • CREATE DEFINER='root'@'localhost' PROCEDURE `get_order_details`(in order_id varchar(100))
7 BEGIN
8 SELECT order_id, p1.person_name as driver, p.person_name as ordered_by, location_name, location_address
9 FROM campus_eats_fall12020.order AS o
10 INNER JOIN
11     person AS p1
12     ON o.driver_id = p1.person_id
13     AND o.order_id = order_id
14     INNER JOIN
15     person AS p
16     ON o.person_id = p.person_id
17     AND o.order_id = order_id
18     INNER JOIN
19     location AS l
20     ON o.location_id = l.location_id
21     AND o.order_id = order_id;
22 END;;
23
24 call get_order_details(1);
```

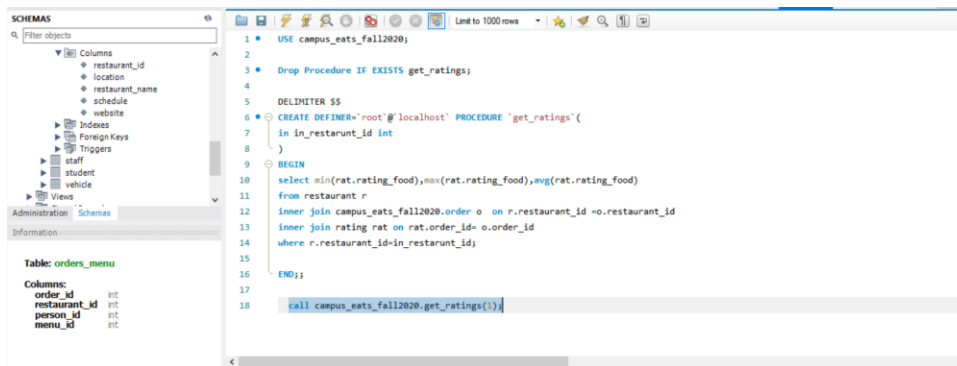
Result:

The screenshot shows the 'Result Grid' of the database IDE. It displays the results of the stored procedure call for order_id 1. The table has five columns: order_id, driver, ordered_by, location_name, and location_address. The first row shows the details for order 1.

order_id	driver	ordered_by	location_name	location_address
1	Keith Turner	Keith Turner	Suite 157	69612 Will Ferry E...

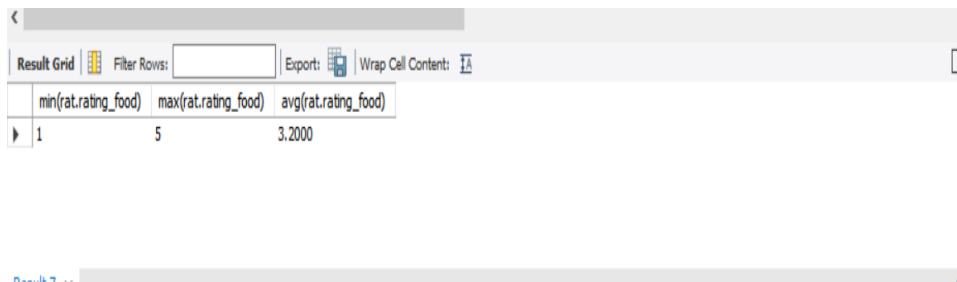
Name: get_ratings

Purpose: To get max,min and avg rating of a restaurant.



```
1 USE campus_eats_fall2020;
2
3 Drop Procedure IF EXISTS get_ratings;
4
5 DELIMITER $$
6 CREATE DEFINER='root'@'localhost' PROCEDURE 'get_ratings'(
7   in_in_restarunt_id int
8 )
9 BEGIN
10  select min(rat.rating_food),max(rat.rating_food),avg(rat.rating_food)
11  from restaurant r
12  inner join campus_eats_fall2020.order o on r.restaurant_id =o.restaurant_id
13  inner join rating rat on rat.order_id= o.order_id
14  where r.restaurant_id=in_restarunt_id;
15
16 END;;
17
18 call campus_eats_fall2020.get_ratings(1);
```

Result:

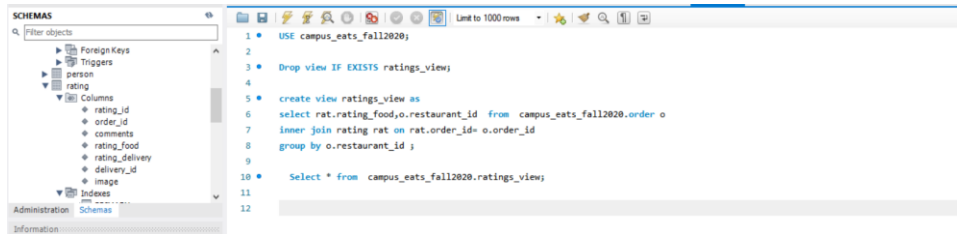


	min(rat.rating_food)	max(rat.rating_food)	avg(rat.rating_food)
1	1	5	3.2000

Views:

Name: ratingsview

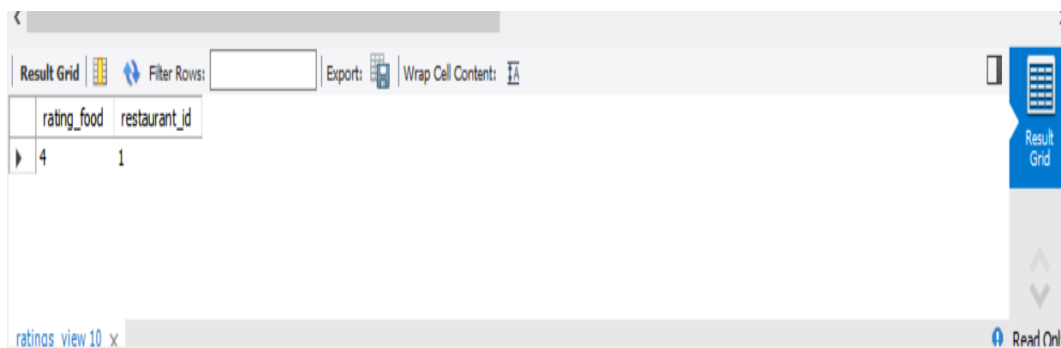
Purpose: to get the ratings



The screenshot shows a database management tool interface. On the left, there is a 'SCHEMAS' pane with a tree view showing 'ForeignKeys', 'Triggers', 'person', 'rating', and 'Columns'. The 'Columns' section is expanded, showing columns like 'rating_id', 'order_id', 'comments', 'rating_food', 'rating_delivery', 'delivery_id', and 'image'. The main pane displays SQL code:

```
1 • USE campus_eats_fall2020;
2
3 • Drop view IF EXISTS ratings_view;
4
5 • create view ratings_view as
6   select rat.rating_food,o.restaurant_id from campus_eats_fall2020.order o
7   inner join rating rat on rat.order_id= o.order_id
8   group by o.restaurant_id ;
9
10 • Select * from campus_eats_fall2020.ratings_views;
11
12
```

Result:



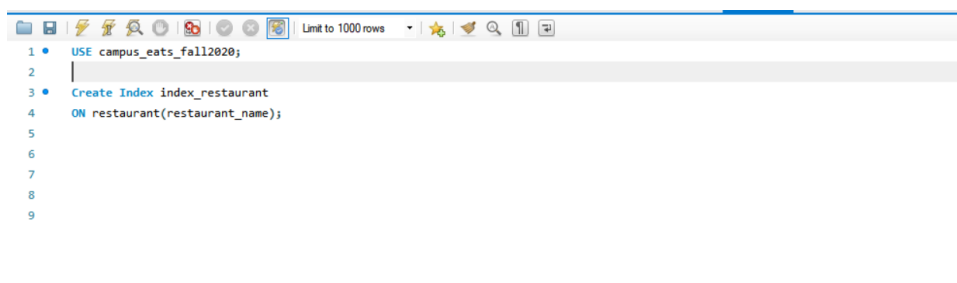
The screenshot shows a database management tool interface. At the top, there is a toolbar with buttons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar, there is a table with two columns: 'rating_food' and 'restaurant_id'. The table contains one row with the values '4' and '1' respectively. On the right side, there is a 'Result Grid' button and a 'Read Only' status indicator.

rating_food	restaurant_id
4	1

Index:

Name: index_restaurant

Purpose: To improve the query performance, we have indexed the restaurant_name since it will be used multiple times while placing an order



The screenshot shows a database management tool interface. The main pane displays SQL code:

```
1 • USE campus_eats_fall2020;
2
3 • Create Index index_restaurant
4   ON restaurant(restaurant_name);
5
6
7
8
9
```

Name: index_menu_item

Purpose: To improve the query performance, we have indexed the menu_item column since it will be used multiple times while placing an order

```
USE campus_eats_fall2020;  
  
Create Index index_menu_item  
ON menu_item(name);
```