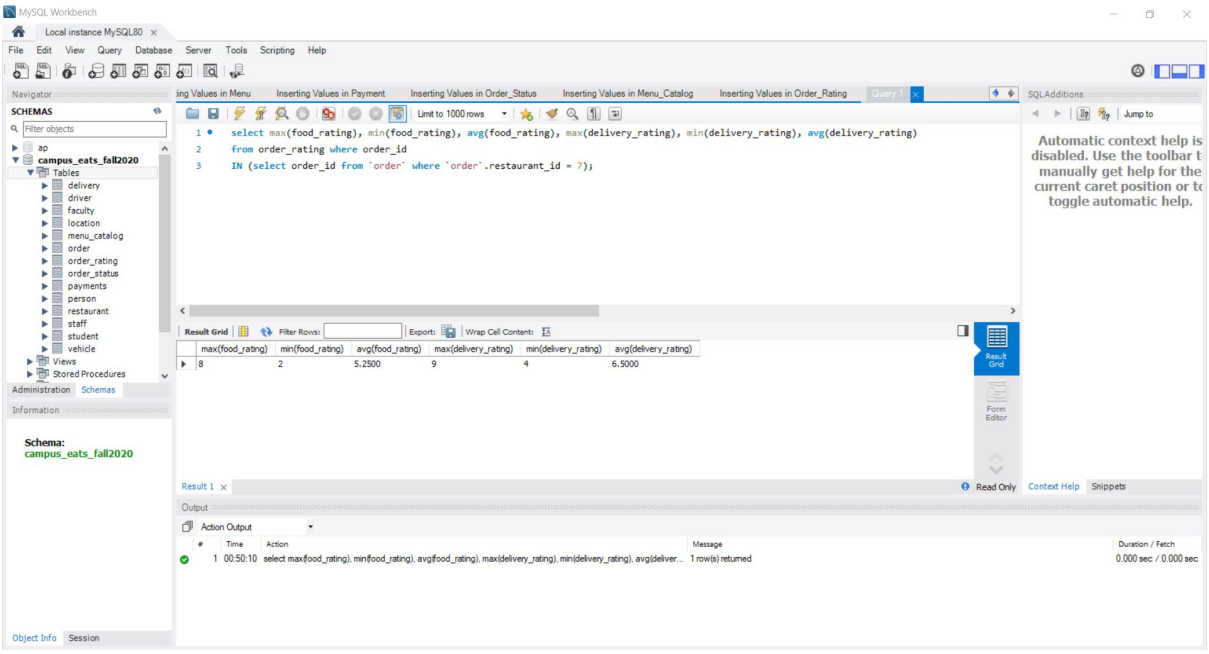
Deliverable 3

Create queries according to those specified in Deliverable 3 in Blackboard:

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

Script

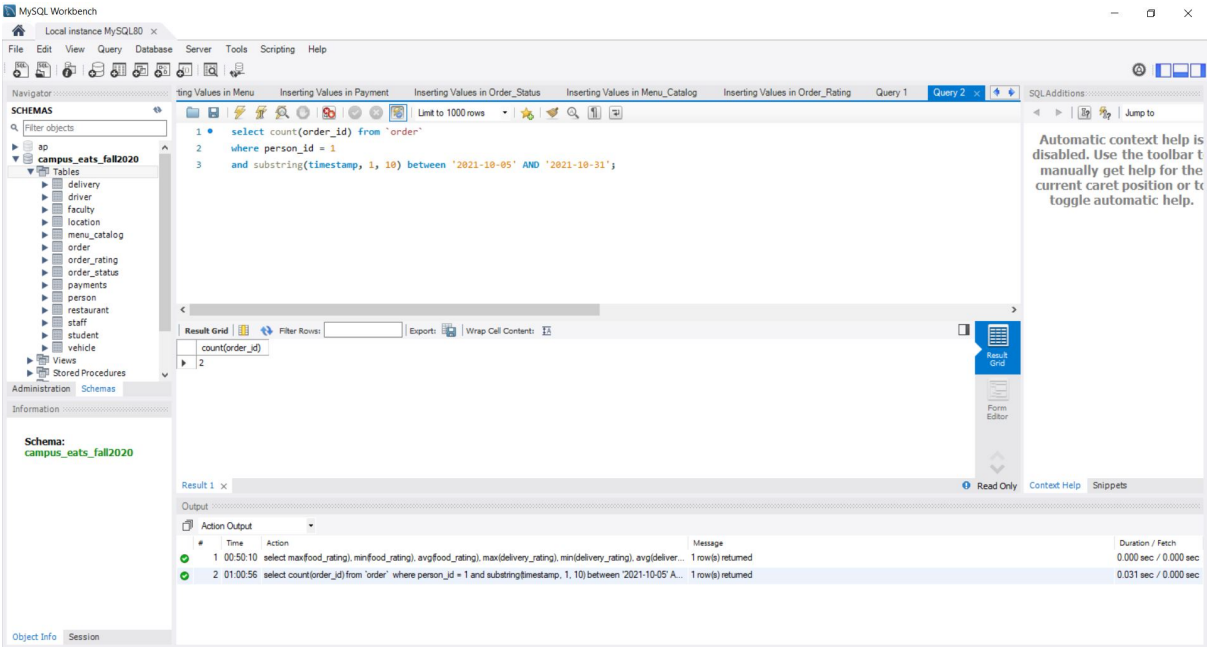
```
select max(food_rating), min(food_rating), avg(food_rating),
max(delivery_rating), min(delivery_rating), avg(delivery_rating)
from order_rating where order_id
IN (select order_id from `order` where `order`.restaurant_id = 7);
```



b) display a count of the orders made by a customer for a specified date range when given a customer id

Script

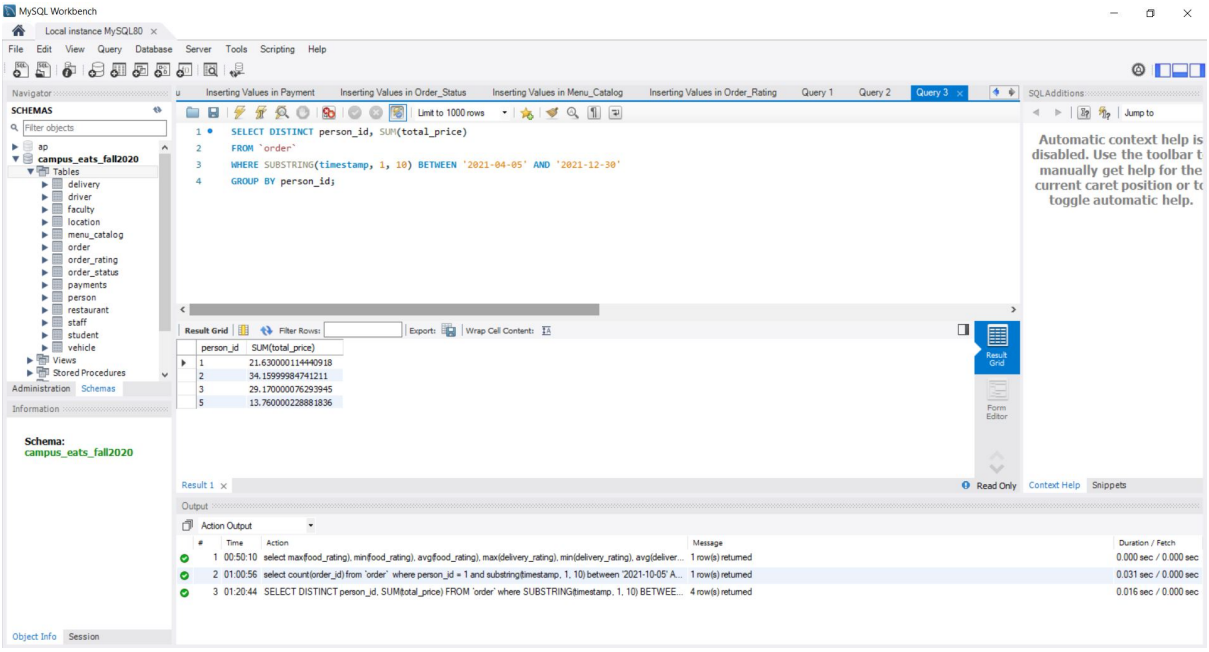
```
select count(order_id) from `order`
where person_id = 1
and substring(timestamp, 1, 10) between '2021-10-05' AND '2021-10-31';
```



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

Script

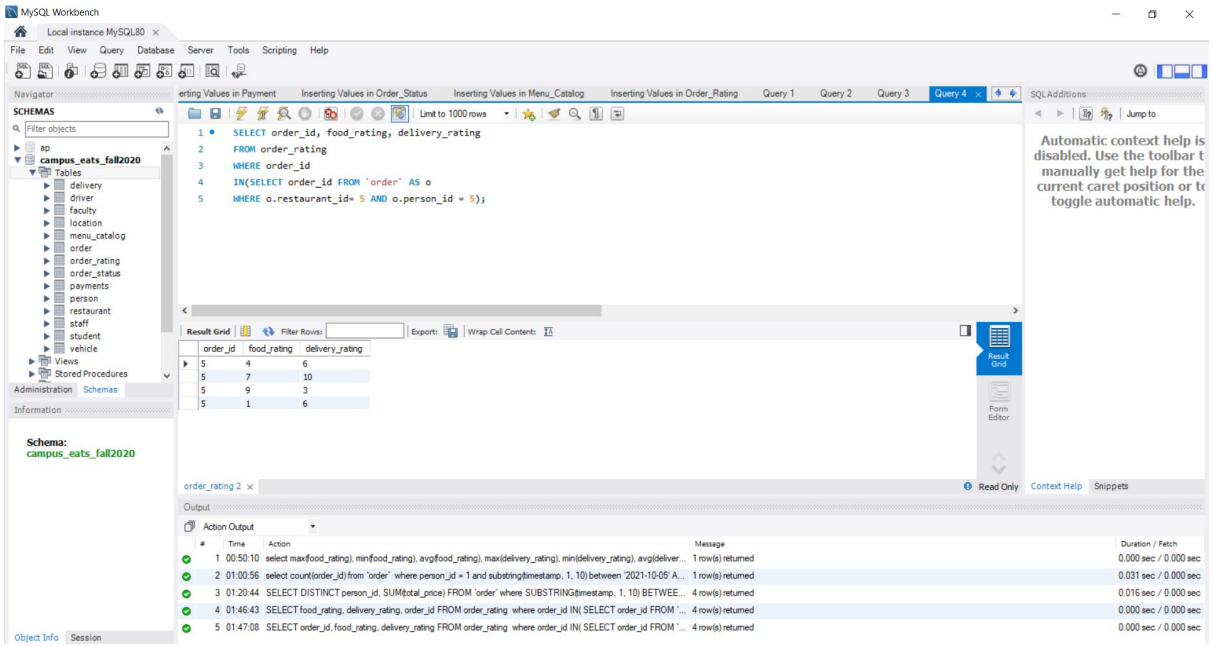
```
SELECT DISTINCT person_id, SUM(total_price)
FROM `order`
WHERE SUBSTRING(timestamp, 1, 10) BETWEEN '2021-04-05' AND '2021-12-30'
GROUP BY person_id;
```



d) display a particular customer’s rating for a restaurant

Script

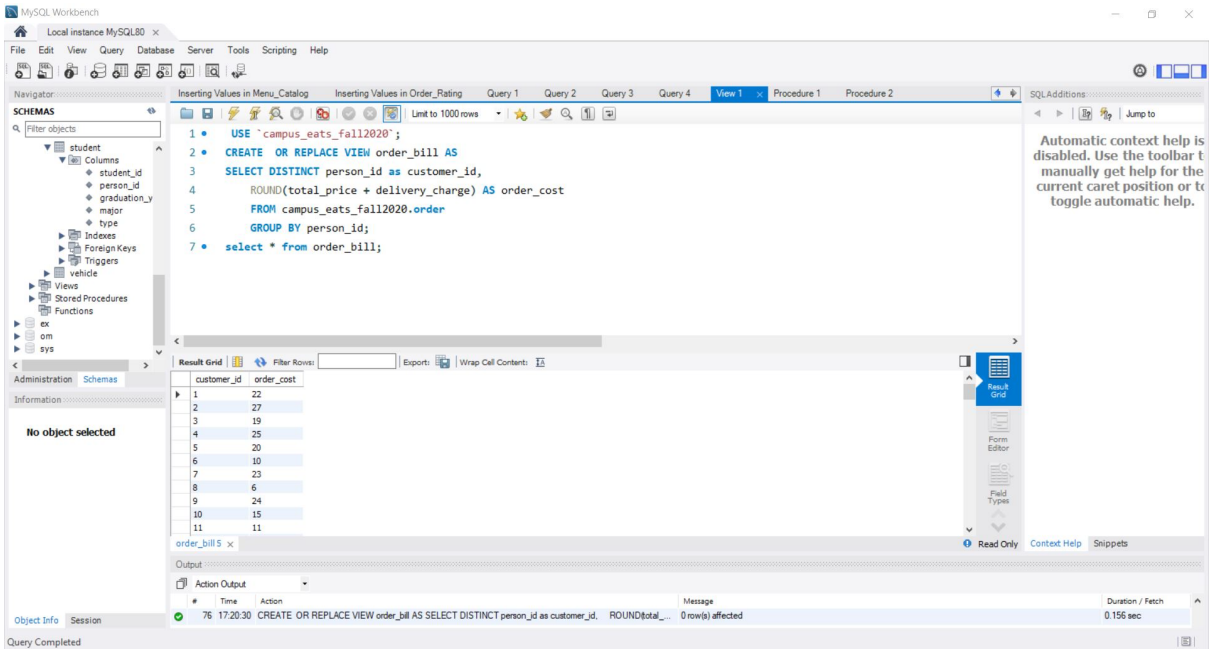
```
SELECT order_id, food_rating, delivery_rating
FROM order_rating
WHERE order_id
IN(SELECT order_id FROM `order` AS o
WHERE o.restaurant_id= 5 AND o.person_id = 5);
```



e) Have one of the above requirements represented in a View

View

```
USE `campus_eats_fall2020`;
CREATE  OR REPLACE VIEW order_bill AS
SELECT DISTINCT person_id as customer_id,
    ROUND(total_price + delivery_charge) AS order_cost
    FROM campus_eats_fall2020.order
    GROUP BY person_id;
select * from order_bill;
```



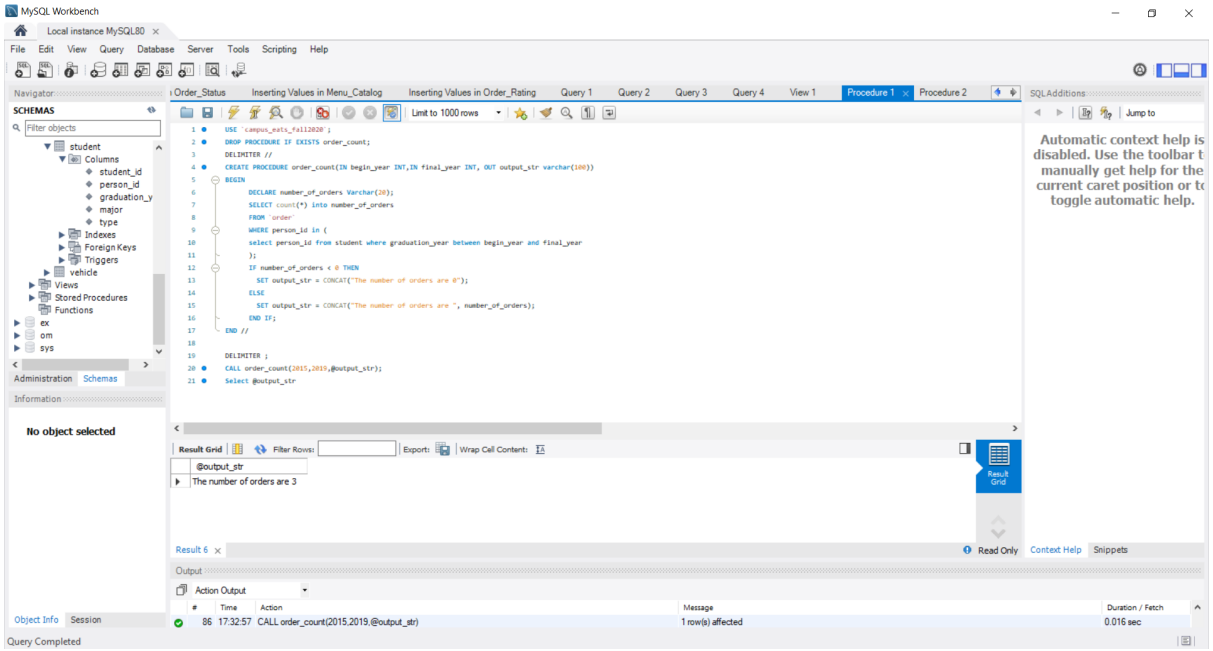
f) Have one of the above requirements represented in a Stored Procedure

Procedure 1

```
USE `campus_eats_fall2020`;
DROP PROCEDURE IF EXISTS order_count;
DELIMITER //
CREATE PROCEDURE order_count(IN begin_year INT,IN final_year INT, OUT output_str varchar(100))
BEGIN
	DECLARE number_of_orders Varchar(20);
	SELECT count(*) into number_of_orders
	FROM `order`
	WHERE person_id in (
	select person_id from student where graduation_year between begin_year and final_year
	);
	IF number_of_orders < 0 THEN
		SET output_str = CONCAT("The number of orders are 0");
	ELSE
		SET output_str = CONCAT("The number of orders are ", number_of_orders);
	END IF;
END //

DELIMITER ;
```

CALL order_count(2015,2019,@output_str);

Select @output_str



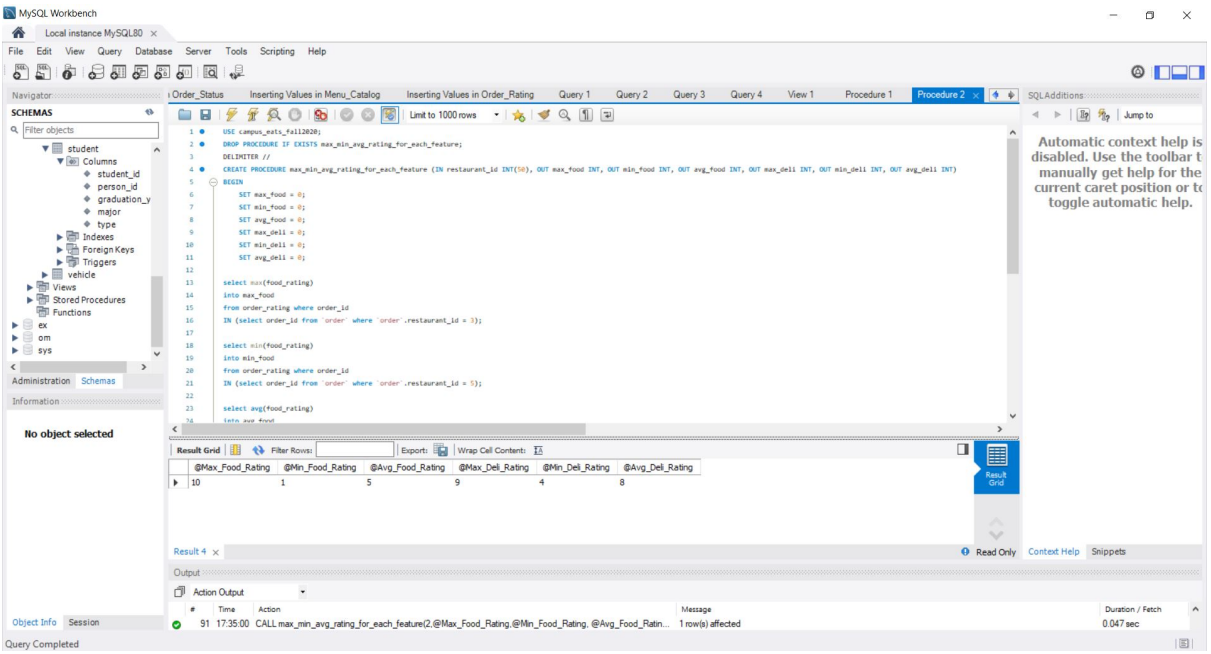
Procedure 2

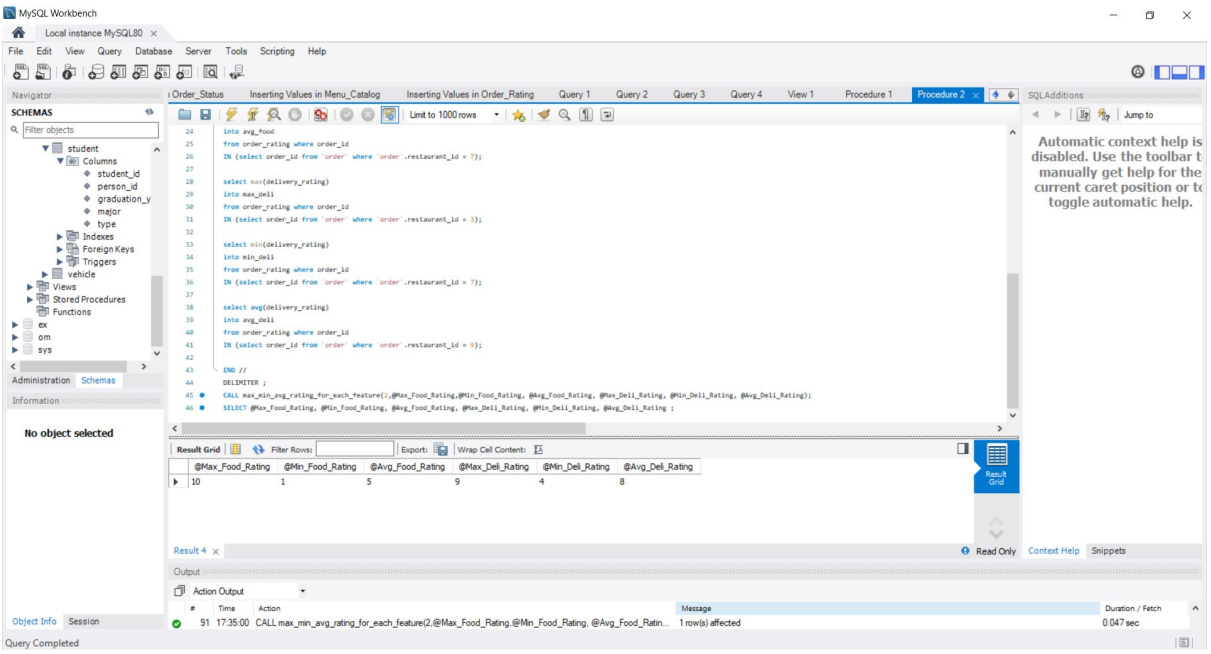
```
USE campus_eats_fall2020;

DROP PROCEDURE IF EXISTS
max_min_avg_rating_for_each_feature;

DELIMITER //

CREATE PROCEDURE max_min_avg_rating_for_each_feature (IN
restaurant_id INT(50), OUT max_food INT, OUT min_food INT,
OUT avg_food INT, OUT max_deli INT, OUT min_deli INT, OUT
avg_deli INT)

BEGIN

    SET max_food = 0;

    SET min_food = 0;

    SET avg_food = 0;

    SET max_deli = 0;

    SET min_deli = 0;

    SET avg_deli = 0;


select max(food_rating)

into max_food

from order_rating where order_id

IN (select order_id from `order` where `order`.restaurant_id = 3);


select min(food_rating)

into min_food

from order_rating where order_id

IN (select order_id from `order` where `order`.restaurant_id = 5);
```

```
select avg(food_rating)
into avg_food
from order_rating where order_id
IN (select order_id from `order` where `order`.restaurant_id = 7);

select max(delivery_rating)
into max_deli
from order_rating where order_id
IN (select order_id from `order` where `order`.restaurant_id = 3);

select min(delivery_rating)
into min_deli
from order_rating where order_id
IN (select order_id from `order` where `order`.restaurant_id = 7);

select avg(delivery_rating)
into avg_deli
from order_rating where order_id
IN (select order_id from `order` where `order`.restaurant_id = 9);



END //
DELIMITER ;
CALL
max_min_avg_rating_for_each_feature(2,@Max_Food_Rating,@Mi
```

n_Food_Rating, @Avg_Food_Rating, @Max_Deli_Rating, @Min_Deli_Rating, @Avg_Deli_Rating);

SELECT @Max_Food_Rating, @Min_Food_Rating, @Avg_Food_Rating, @Max_Deli_Rating, @Min_Deli_Rating, @Avg_Deli_Rating ;

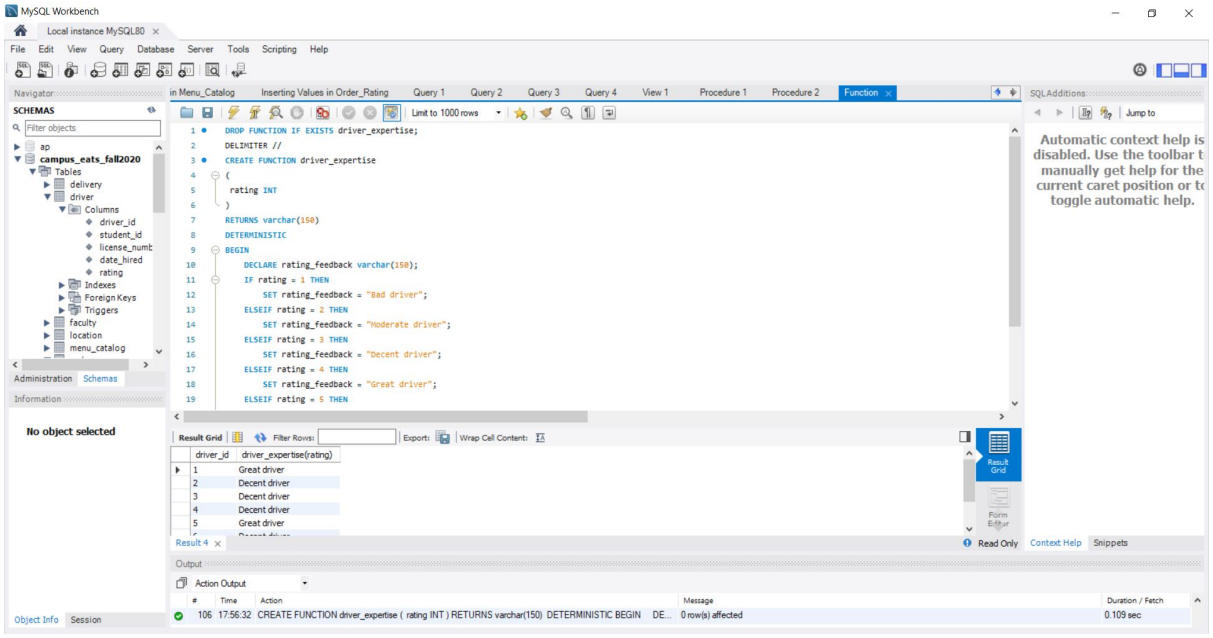


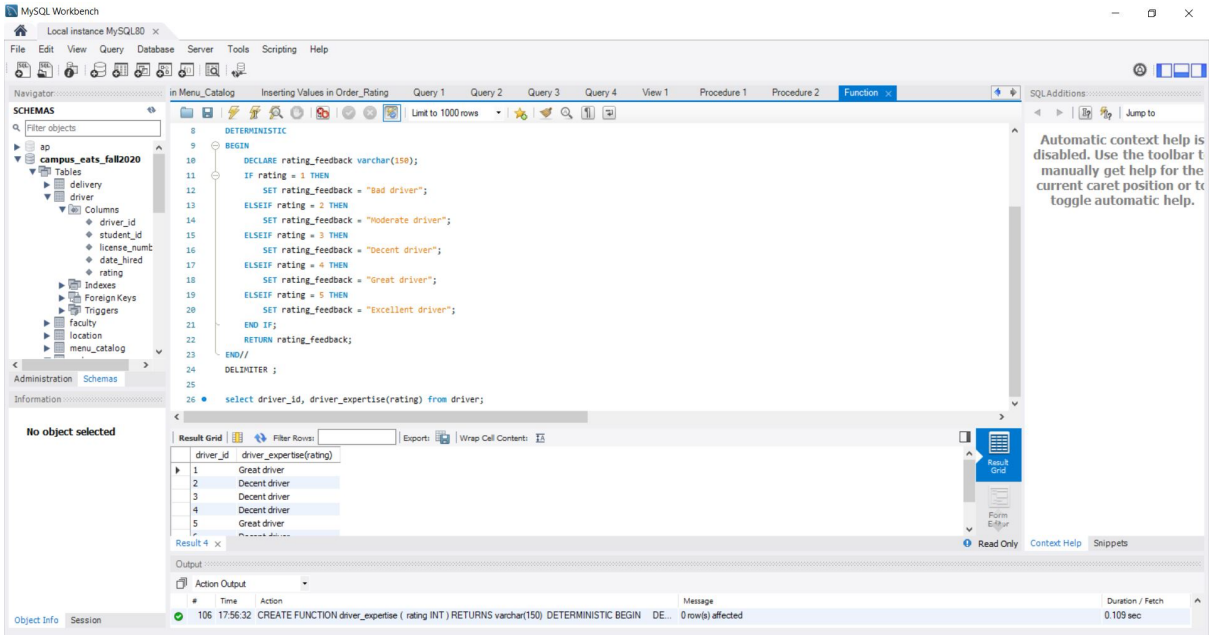


Function

```
DROP FUNCTION IF EXISTS driver_expertise;
DELIMITER //
CREATE FUNCTION driver_expertise
(
 rating INT
)
RETURNS varchar(150)
DETERMINISTIC
BEGIN
   DECLARE rating_feedback varchar(150);
   IF rating = 1 THEN
            SET rating_feedback = "Bad driver";
      ELSEIF rating = 2 THEN
            SET rating_feedback = "Moderate driver";
      ELSEIF rating = 3 THEN
            SET rating_feedback = "Decent driver";
      ELSEIF rating = 4 THEN
            SET rating_feedback = "Great driver";
      ELSEIF rating = 5 THEN
            SET rating_feedback = "Excellent driver";
      END IF;
   RETURN rating_feedback;
END//
DELIMITER ;
```

select driver_id, driver_expertise(rating) from driver;





Index

CREATE TABLE IF NOT EXISTS
`campus_eats_fall2020`.`order_rating` (`rating_id` INT NOT NULL,

`order_id` INT NOT NULL,

`food_rating` INT NULL,

`delivery_rating` INT NULL,

`comments` VARCHAR(200) NULL,

PRIMARY KEY (`rating_id`),

INDEX `order_id_indx` (`order_id` ASC) VISIBLE, CONSTRAINT `order_id`

FOREIGN KEY (`order_id`)

REFERENCES `campus_eats_fall2020`.`order` (`order_id`))