

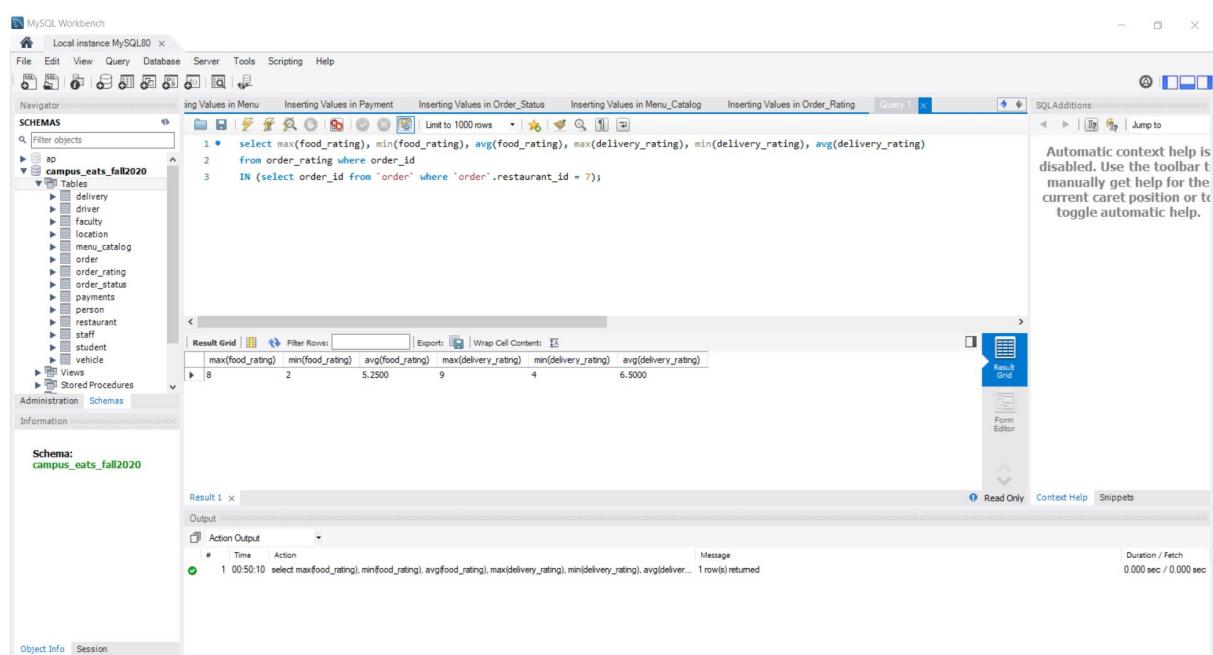
Deliverable 3

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);
```



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `Campus_eats_fall2020` with its tables: delivery, driver, faculty, location, menu_catalog, order, order_rating, order_status, payments, person, restaurant, staff, student, vehicle.
- Query Editor:** Contains the SQL query:

```
1 • select max(food_rating), min(food_rating), avg(food_rating), max(delivery_rating), min(delivery_rating), avg(delivery_rating)
2   from order_rating where order_id
3   IN (select order_id from `order` where `order`.restaurant_id = 7);
```
- Result Grid:** Displays the results of the query:

	max(food_rating)	min(food_rating)	avg(food_rating)	max(delivery_rating)	min(delivery_rating)	avg(delivery_rating)
1	8	2	5.2500	9	4	6.5000
- Action Output:** Shows the log of the executed action:

```
# Time Action Message Duration / Fetch
1 00:50:10 select max(food_rating), min(food_rating), avg(food_rating), max(delivery_rating), min(delivery_rating), avg(delivery... 1 row(s) returned 0.000 sec / 0.000 sec
```

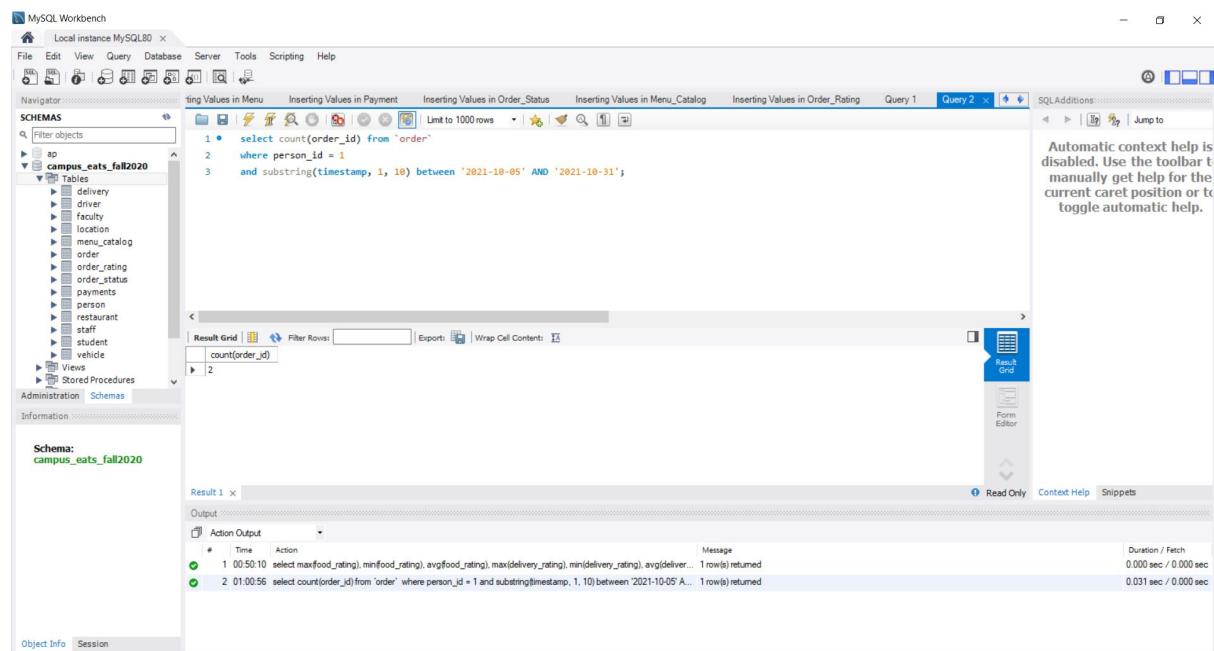
b) display a count of the orders made by a customer for a specified date range when given a customer id

Script

```
select count(order_id) from `order`
```

```
where person_id = 1
```

```
and substring(timestamp, 1, 10) between '2021-10-05' AND '2021-10-31';
```



The screenshot shows the MySQL Workbench interface with the following details:

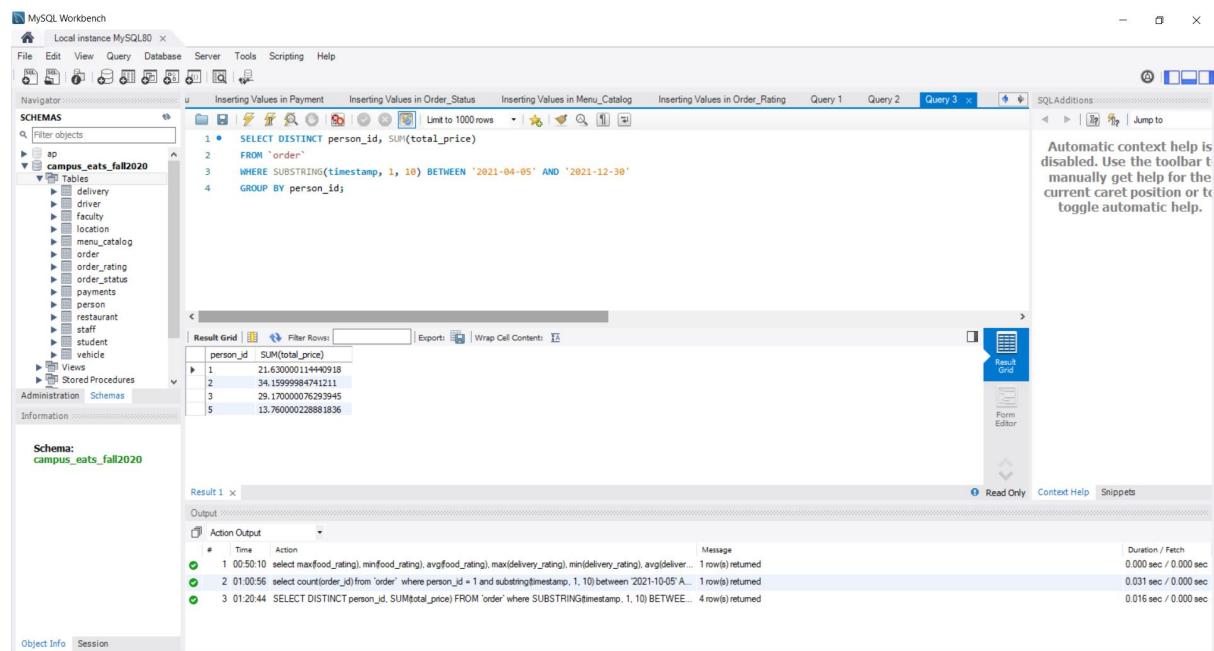
- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Includes icons for New Connection, Open Connection, Save, Print, Copy, Paste, Find, Replace, and others.
- Schemas:** Local instance MySQL80, campus_eats_fall2020.
- Tables:** delivery, driver, faculty, location, menu_catalog, order, order_rating, order_status, payments, person, restaurant, staff, student, vehicle.
- Query Editor:** Contains three queries:
 1. select count(order_id) from `order`;
 2. where person_id = 1
 3. and substring(timestamp, 1, 10) between '2021-10-05' AND '2021-10-31';
- Result Grid:** Shows the result of the first query: count(order_id) = 2.
- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	00:50:10	select max(food_rating), min(food_rating), avg(food_rating), max(delivery_rating), min(delivery_rating), avg(deliver...	1 row(s) returned	0.000 sec / 0.000 sec
2	01:00:56	select count(order_id) from `order` where person_id = 1 and substring(timestamp, 1, 10) between '2021-10-05' A...	1 row(s) returned	0.031 sec / 0.000 sec

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;
```



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the SQL query:

```
1 • SELECT DISTINCT person_id, SUM(total_price)
2   FROM `order`
3   WHERE SUBSTRING(timestamp, 1, 10) BETWEEN '2021-04-05' AND '2021-12-30'
4   GROUP BY person_id;
```
- Result Grid:** Displays the results of the query:

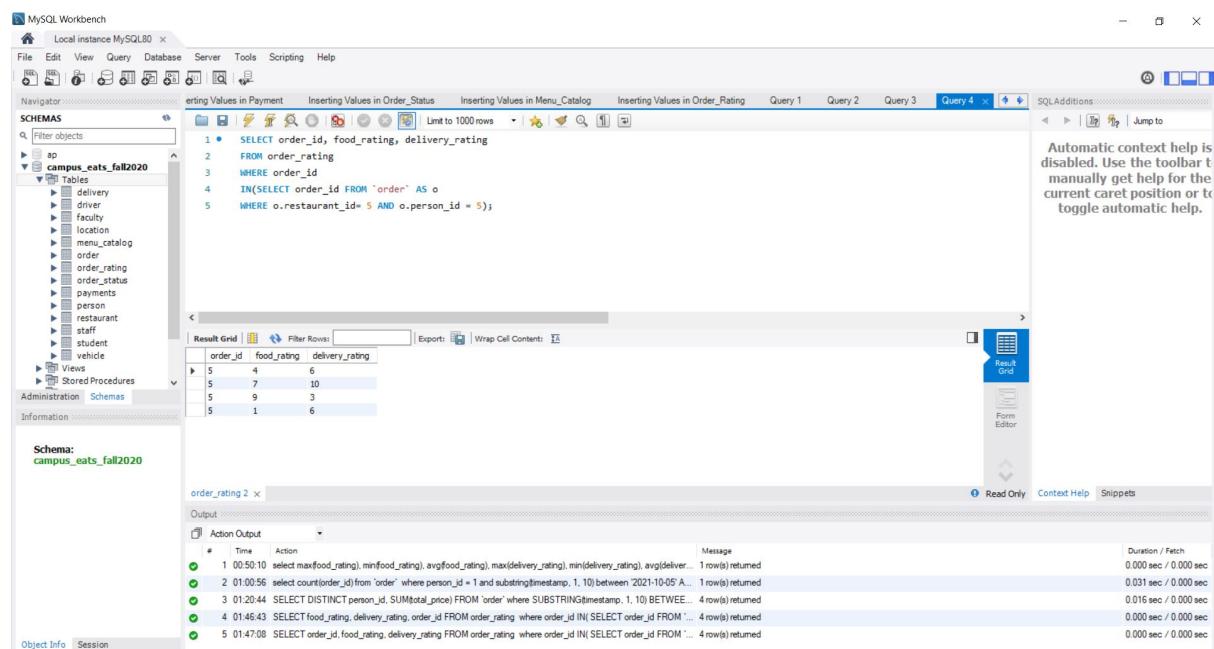
person_id	SUM(total_price)
1	21.63000011440918
2	34.15999984741211
3	29.170000076293945
5	13.760000228881836
- Action Output:** Shows the log of actions taken during the query execution:

#	Time	Action	Message	Duration / Fetch
1	00:50:10	select max(food_rating), min(food_rating), avg(food_rating), max(delivery_rating), min(delivery_rating), avg(delivery_rating);	1 row(s) returned	0.000 sec / 0.000 sec
2	01:00:56	select count(order_id) from `order` where person_id = 1 and substring(timestamp, 1, 10) between '2021-10-05' AND '2021-10-31';	1 row(s) returned	0.031 sec / 0.000 sec
3	01:20:44	SELECT DISTINCT person_id, SUM(total_price) FROM `order` WHERE SUBSTRING(timestamp, 1, 10) BETWEEN '2021-04-05' AND '2021-12-30';	4 row(s) returned	0.016 sec / 0.000 sec

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);
```



The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Includes icons for New Connection, Open Connection, Save, Print, Copy, Paste, Find, Replace, and others.
- Schemas:** Shows the current database schema: campus_eats_fall2020, which contains tables like delivery, driver, faculty, location, menu_catalog, order, order_rating, order_status, payments, person, restaurant, staff, student, vehicle.
- Query Editor:** Contains the SQL query:1 • SELECT order_id, food_rating, delivery_rating
2 FROM order_rating
3 WHERE order_id
4 IN(SELECT order_id FROM `order` AS o
5 WHERE o.restaurant_id= 5 AND o.person_id = 5);
- Result Grid:** Displays the query results in a grid format:| order_id | food_rating | delivery_rating |
| --- | --- | --- |
| 5 | 4 | 6 |
| 5 | 7 | 10 |
| 5 | 9 | 3 |
| 5 | 1 | 6 |
- Output Tab:** Shows the execution log:# 1 00:50:10 select max(food_rating), min(food_rating), avg(food_rating), max(delivery_rating), min(delivery_rating), avg(delivery_rating) returned
2 01:00:56 select count(order_id) from `order` where person_id >= 1 and substring(timestamp, 1, 10) between '2021-10-05 A...' 1 row(s) returned
3 01:20:44 SELECT DISTINCT person_id, SUM(total_price) FROM `order` where SUBSTRING(timestamp, 1, 10) BETWEEN '2021-10-05 A...' 4 row(s) returned
4 01:46:43 SELECT food_rating, delivery_rating, order_id FROM order_rating where order_id IN (SELECT order_id FROM `order` AS o WHERE o.restaurant_id= 5 AND o.person_id = 5); 4 row(s) returned
5 01:47:08 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); 4 row(s) returned

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;
```

The screenshot shows the MySQL Workbench interface with the following details:

- File Menu:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, showing the 'campus_eats_fall2020' database structure with tables like student, vehicle, etc.
- SQL Editor:** Displays the SQL code for creating the view:

```
1 •  USE `campus_eats_fall2020`;
2 •  CREATE OR REPLACE VIEW order_bill AS
3     SELECT DISTINCT person_id as customer_id,
4         ROUND(total_price + delivery_charge) AS order_cost
5     FROM campus_eats_fall2020.order
6     GROUP BY person_id;
7 •  select * from order_bill;
```

- Result Grid:** Shows the results of the query, displaying 11 rows of data:

customer_id	order_cost
1	22
2	27
3	19
4	25
5	20
6	10
7	23
8	6
9	24
10	15
11	11

- Output:** Shows the execution message: "17:20:30 CREATE OR REPLACE VIEW order_bill AS SELECT DISTINCT person_id as customer_id, ROUND(total_... 0 row(s) affected".

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

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons.
- Navigator:** Local instance MySQL80, Schemas (student), Tables (Order_Status, Inserting Values in Menu_Catalog, Inserting Values in Order_Rating, Query 1, Query 2, Query 3, Query 4, View 1, Procedure 1, Procedure 2).
- Procedure 1 Tab:** Contains the stored procedure code:

```
1 USE campus_reta_fall2020 ;
2 DROP PROCEDURE IF EXISTS order_count;
3 DELIMITER //
4 CREATE PROCEDURE order_count(IN begin_year INT, IN final_year INT, OUT output_str varchar(100))
5 BEGIN
6     DECLARE number_of_orders Varchar(20);
7     SELECT count(*) INTO number_of_orders
8     FROM order;
9     WHERE person_id IN (
10         select person_id from student where graduation_year between begin_year and final_year
11     );
12     IF number_of_orders < 0 THEN
13         SET output_str = CONCAT("The number of orders are ");
14     ELSE
15         SET output_str = CONCAT("The number of orders are ", number_of_orders);
16     END IF;
17 END //
18 //
19 DELIMITER ;
20 CALL order_count(2015,2019,@output_str);
21 Select @output_str
```
- Result Grid:** Shows the output of the stored procedure call: "@output_str" and "The number of orders are 3".
- Result 6 Tab:** Shows the history of the stored procedure call: Action Output, # 86, Time 17:32:57, Action CALL order_count(2015,2019,@output_str), Message 1 row(s) affected, Duration / Fetch 0.016 sec.
- Information:** No object selected.
- Help:** Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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);
```

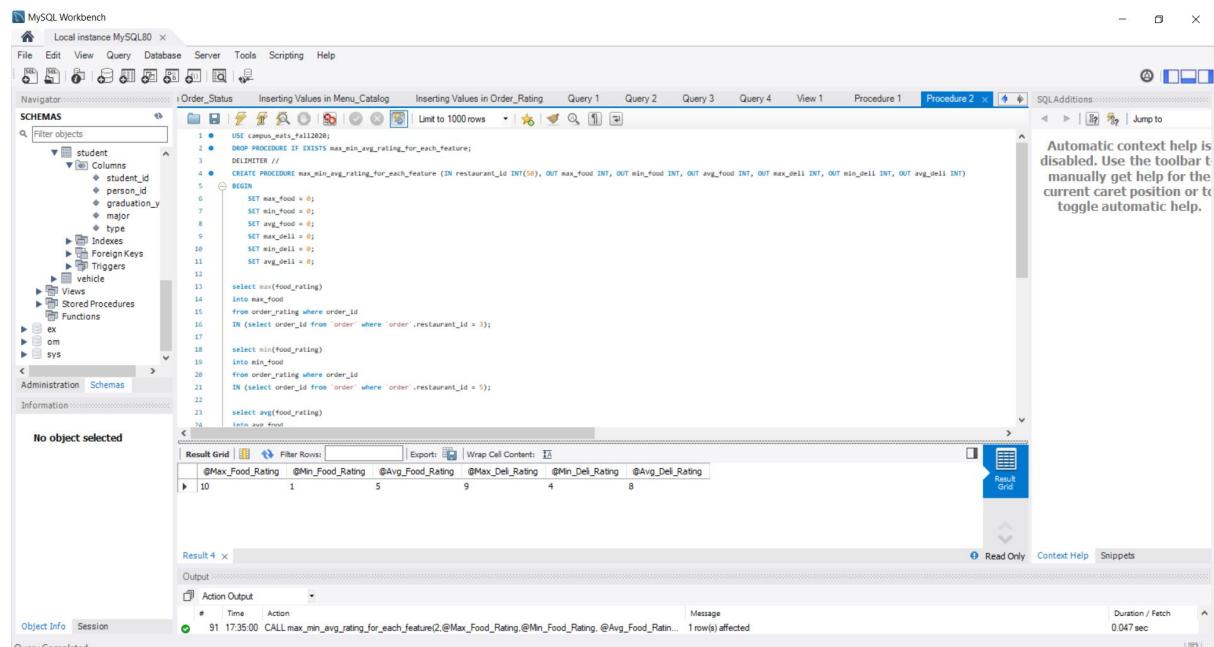
```
select min(delivery_rating)
into min_deli
from order_rating where order_id
IN (select order_id from `order` where `order`.restaurant_id = 7);
```

```
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 ;`



```

USE carpeta_exams_fall2020;
DELIMITER //
CREATE PROCEDURE max_min_avg_rating_for_each_feature (IN restaurant_id INT(10), 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 = 1);

    select min(food_rating)
    into min_food
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);

    select avg(food_rating)
    into avg_food
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);

    select max(delivery_rating)
    into max_deli
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);

    select min(delivery_rating)
    into min_deli
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);

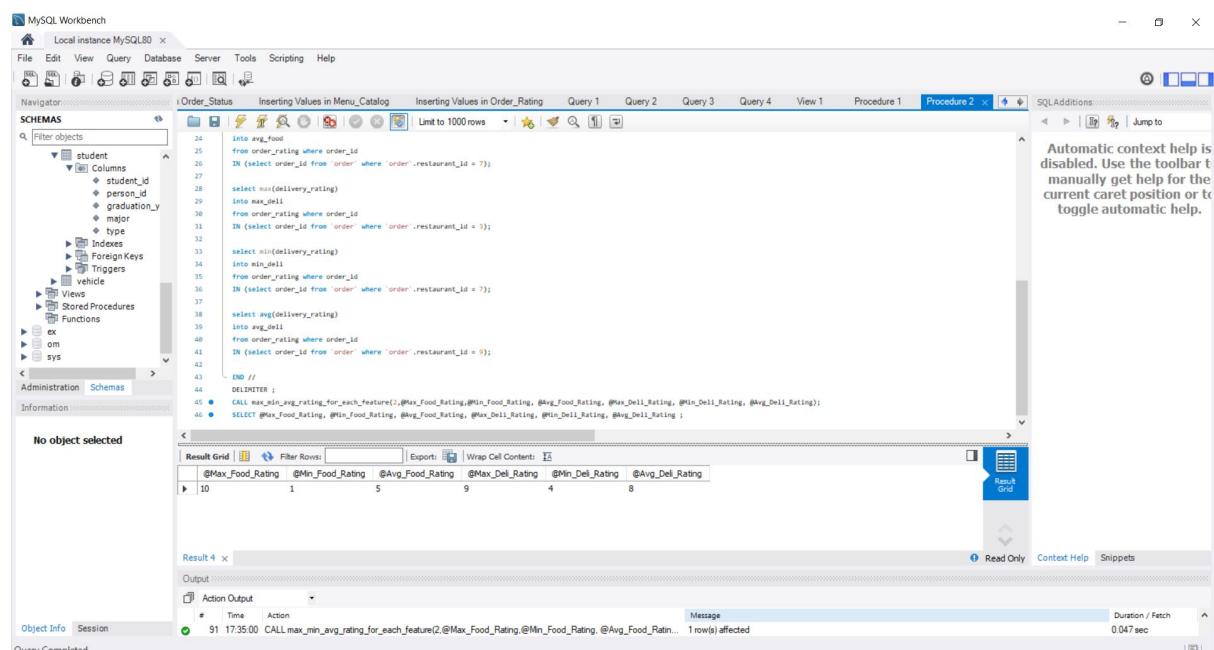
    select avg(delivery_rating)
    into avg_deli
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);
END //
DELIMITER ;
CALL max_min_avg_rating_for_each_feature(2,@Max_Food_Rating,@Min_Food_Rating,@Avg_Food_Rating,@Max_Deli_Rating,@Min_Deli_Rating,@Avg_Deli_Rating);
SELECT @Max_Food_Rating, @Min_Food_Rating, @Avg_Food_Rating, @Max_Deli_Rating, @Min_Deli_Rating, @Avg_Deli_Rating;

```

	@Max_Food_Rating	@Min_Food_Rating	@Avg_Food_Rating	@Max_Deli_Rating	@Min_Deli_Rating	@Avg_Deli_Rating
10	1	5	9	4	8	

Action Output

#	Time	Action	Message	Duration / Fetch
91	17:35:00	CALL max_min_avg_rating_for_each_feature(2,@Max_Food_Rating,@Min_Food_Rating,@Avg_Food_Rating,@Max_Deli_Rating,@Min_Deli_Rating,@Avg_Deli_Rating);	1 row(s) affected	0.04 sec



```

USE carpeta_exams_fall2020;
DELIMITER //
CREATE PROCEDURE max_min_avg_rating_for_each_feature (IN restaurant_id INT(10), 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 = 1);

    select min(food_rating)
    into min_food
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);

    select avg(food_rating)
    into avg_food
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);

    select max(delivery_rating)
    into max_deli
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);

    select min(delivery_rating)
    into min_deli
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);

    select avg(delivery_rating)
    into avg_deli
    from order_rating where order_id
    IN (select order_id from order where order.restaurant_id = 1);
END //
DELIMITER ;
CALL max_min_avg_rating_for_each_feature(2,@Max_Food_Rating,@Min_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;

```

	@Max_Food_Rating	@Min_Food_Rating	@Avg_Food_Rating	@Max_Deli_Rating	@Min_Deli_Rating	@Avg_Deli_Rating
10	1	5	9	4	8	

Action Output

#	Time	Action	Message	Duration / Fetch
91	17:35:00	CALL max_min_avg_rating_for_each_feature(2,@Max_Food_Rating,@Min_Food_Rating,@Avg_Food_Rating,@Max_Deli_Rating,@Min_Deli_Rating,@Avg_Deli_Rating);	1 row(s) affected	0.04 sec

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;

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Tables:** campus_eats_fall2020 (selected), delivery, driver
- Function:** driver_expertise
- Code:**

```
1 • DROP FUNCTION IF EXISTS driver_expertise;
2 • DELIMITER //
3 • CREATE FUNCTION driver_expertise
4 • (
5 •     rating INT
6 • )
7 • RETURNS varchar(150)
8 • DETERMINISTIC
9 • BEGIN
10 •     DECLARE rating_feedback varchar(150);
11 •     IF rating = 1 THEN
12 •         SET rating_feedback = "Bad driver";
13 •     ELSEIF rating = 2 THEN
14 •         SET rating_feedback = "Moderate driver";
15 •     ELSEIF rating = 3 THEN
16 •         SET rating_feedback = "Decent driver";
17 •     ELSEIF rating = 4 THEN
18 •         SET rating_feedback = "Great driver";
19 •     ELSEIF rating = 5 THEN
20 •         SET rating_feedback = "Excellent driver";
21 •     END IF;
22 •     RETURN rating_feedback;
23 • END//
```
- Result Grid:** Shows the output of the function for ratings 1 through 5:

driver_id	driver_expertise(rating)
1	Great driver
2	Decent driver
3	Decent driver
4	Decent driver
5	Great driver
- Action Output:** Shows the creation of the function:

#	Time	Action
106	17:56:32	CREATE FUNCTION driver_expertise (rating INT) RETURNS varchar(150) DETERMINISTIC BEGIN DE... 0 row(s) affected

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Tables:** campus_eats_fall2020 (selected), delivery, driver
- Query:** select driver_id, driver_expertise(rating) from driver;
- Result Grid:** Shows the output of the query for ratings 1 through 5:

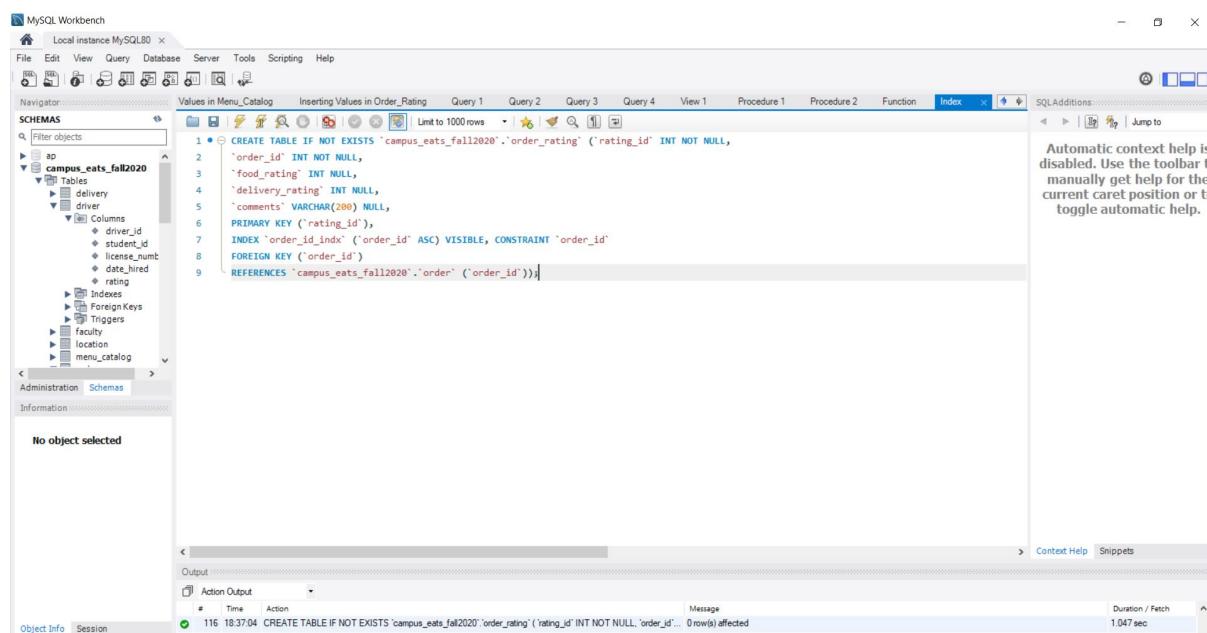
driver_id	driver_expertise(rating)
1	Great driver
2	Decent driver
3	Decent driver
4	Decent driver
5	Great driver
- Action Output:** Shows the creation of the function:

#	Time	Action
106	17:56:32	CREATE FUNCTION driver_expertise (rating INT) RETURNS varchar(150) DETERMINISTIC BEGIN DE... 0 row(s) affected

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_idx`(`order_id` ASC) VISIBLE, CONSTRAINT  
`order_id`  
FOREIGN KEY(`order_id`)  
REFERENCES `campus_eats_fall2020`.`order`(`order_id`))
```



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "campus_eats_fall2020".
- Tables:** The table "order_rating" is selected.
- Query Editor:** The SQL code for creating the table is displayed:

```
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_idx`(`order_id` ASC) VISIBLE, CONSTRAINT  
`order_id`  
FOREIGN KEY(`order_id`)  
REFERENCES `campus_eats_fall2020`.`order`(`order_id`))
```
- Output Tab:** The output shows the execution results:

#	Time	Action	Message	Duration / Fetch
116	18:37:04	CREATE TABLE IF NOT EXISTS `campus_eats_fall2020`.`order_rating`(`rating_id` INT NOT NULL, `order_id`... 0 row(s) affected		1.047 sec