PES UNIVERSITY
Department of Computer Science & Engineering

DBMS - UE20CS301
Mini Project
Online Food Ordering System

Submitted to:
Dr. Geetha D

Associate Professor

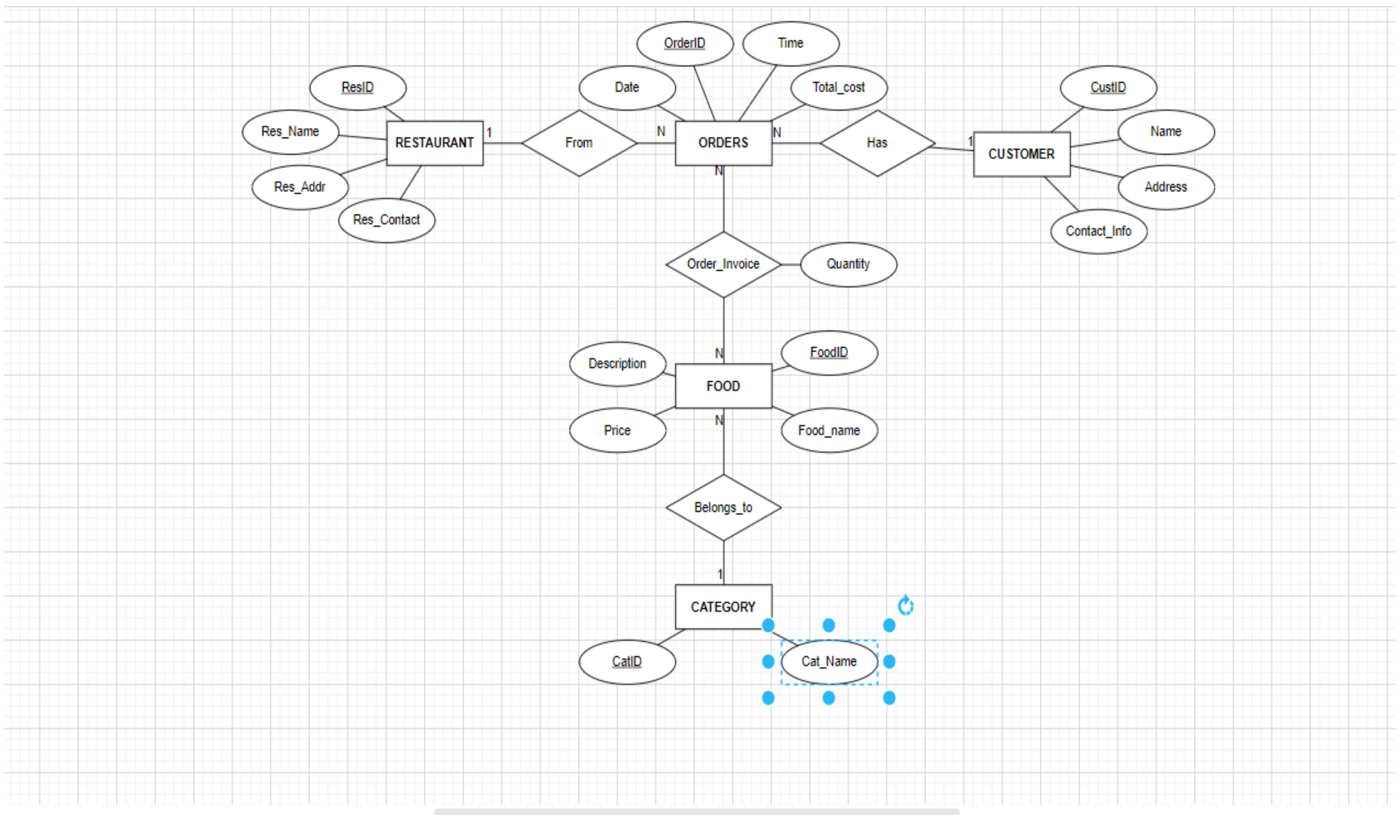
Submitted By:
Name: Shreyas Sai Raman
SRN:PES2UG20CS461
V Semester
Section : G

Table of Contents

1. Short Description and Scope of the Project

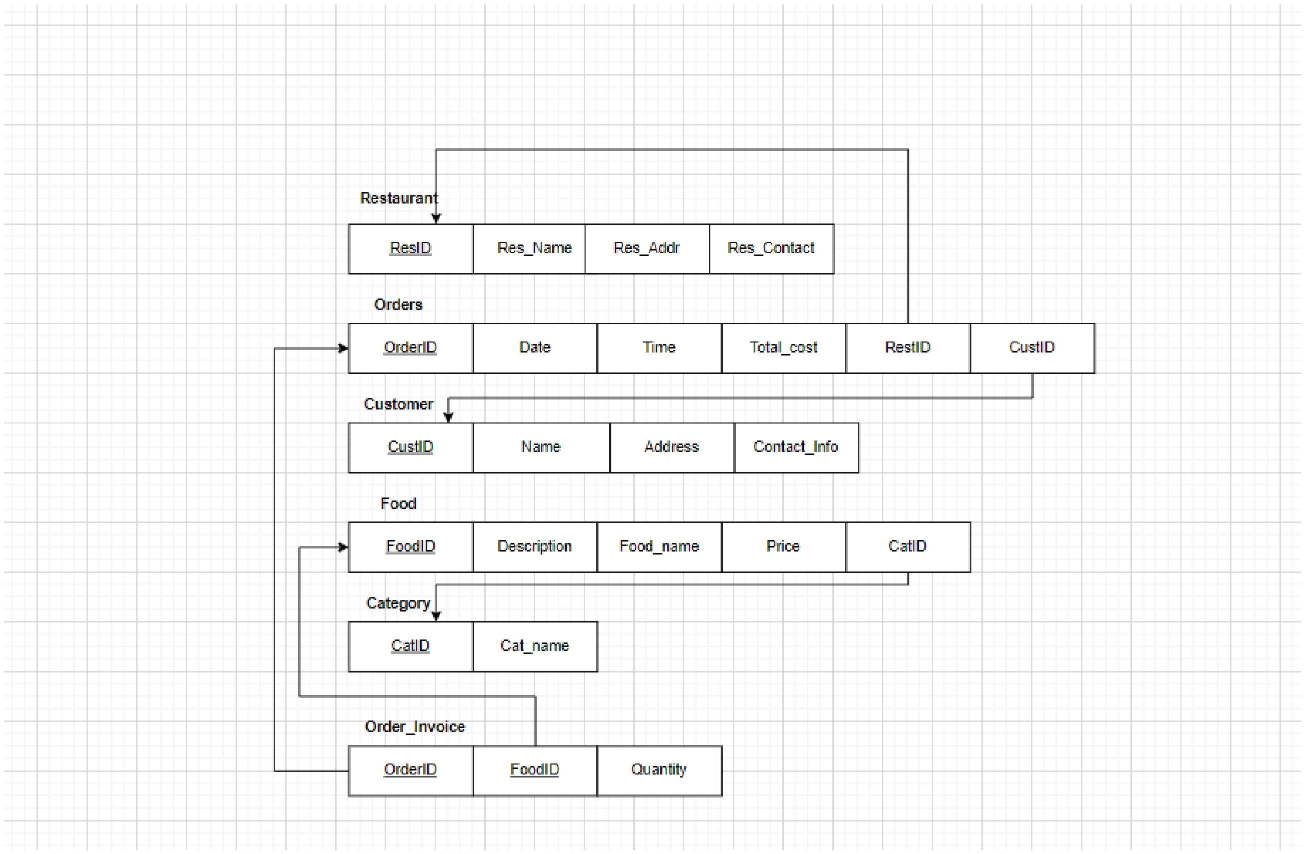
The main objective of online food ordering system is to create a system that allows to order food online. The next important objective is to make the process of ordering quick, easy and convenient. The system is user friendly so that any person using it will not face any difficulties in operating it. The system has facilities that allows users to view menu card, select the category of the food , select the food under the given category in a particular restaurant, adjust the quantity of selected food, select address for food delivery and make the final payment. Data redundancy is reduced and less time consuming which increases the efficiency of the system.

2. ER Diagram



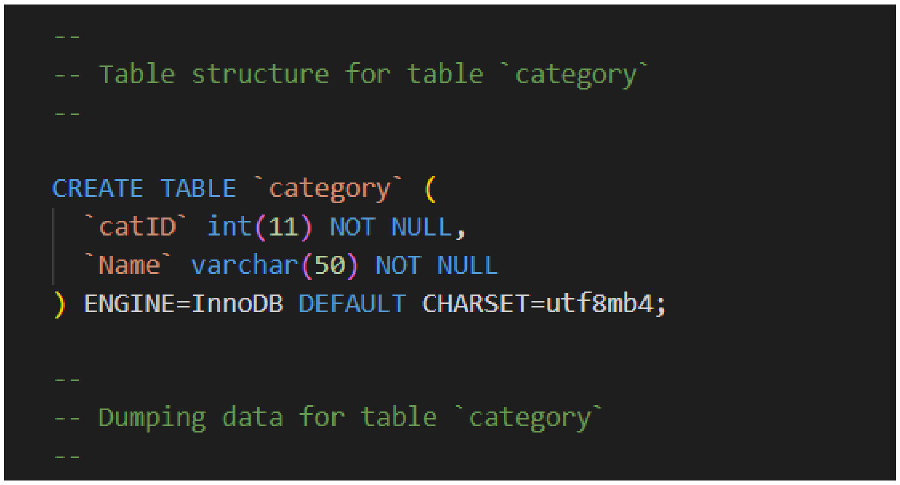
OrderID
Time
ResID
Date
Total_cost
CustID
Res_Name
RESTAURANT
1
From
N
ORDERS
N
Has
1
CUSTOMER
Name
Res_Addr
N
Address
Res_Contact
Contact_Info
Order_Invoice
Quantity
Description
N
FoodID
FOOD
Price
N
Food_name
Belongs_to
1
CATEGORY
CatID
Cat_Name

3. Relational Schema



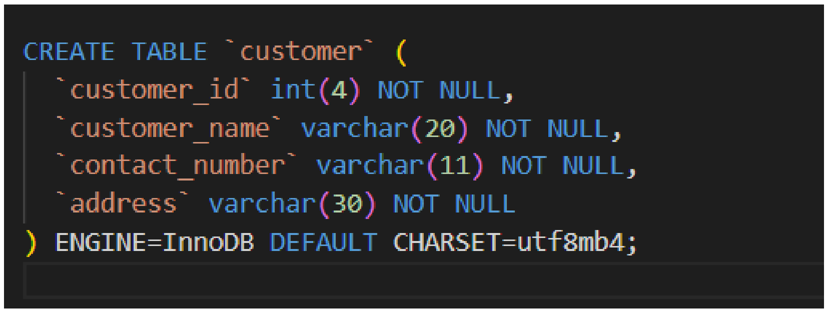
4. DDL statements - Building the database

1) To create table ‘category’



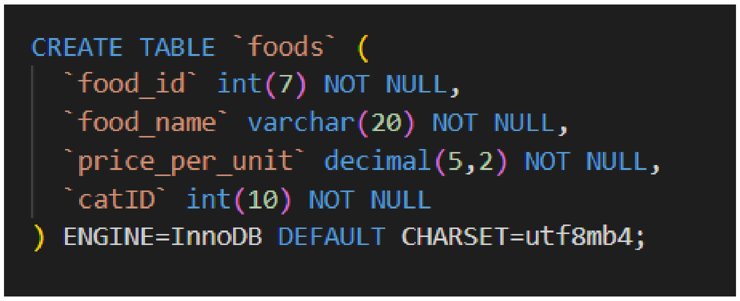
```
--
-- Table structure for table `category`
--

CREATE TABLE `category` (
  `catID` int(11) NOT NULL,
  `Name` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Dumping data for table `category`
--
```

2) To create table ‘customer’



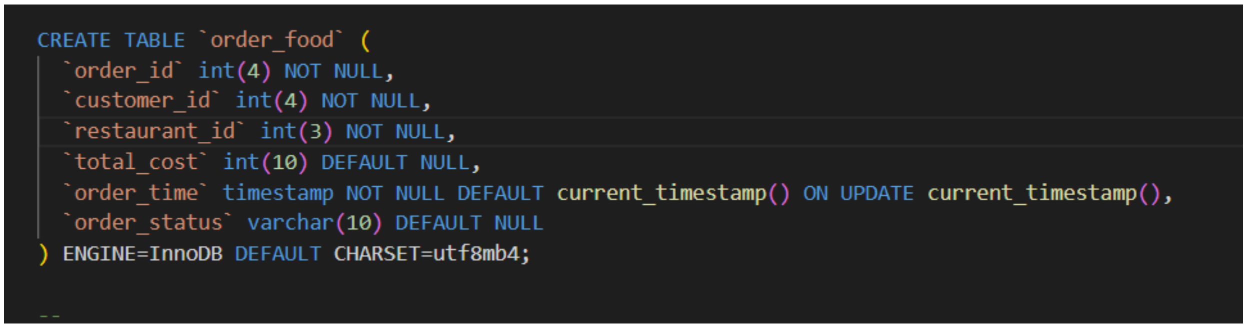
```
CREATE TABLE `customer` (
  `customer_id` int(4) NOT NULL,
  `customer_name` varchar(20) NOT NULL,
  `contact_number` varchar(11) NOT NULL,
  `address` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

3) To create table ‘foods’



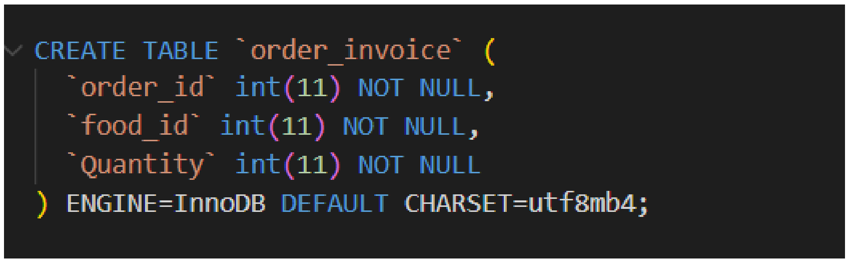
```
CREATE TABLE `foods` (
  `food_id` int(7) NOT NULL,
  `food_name` varchar(20) NOT NULL,
  `price_per_unit` decimal(5,2) NOT NULL,
  `catID` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

4) To create table ‘order_food’



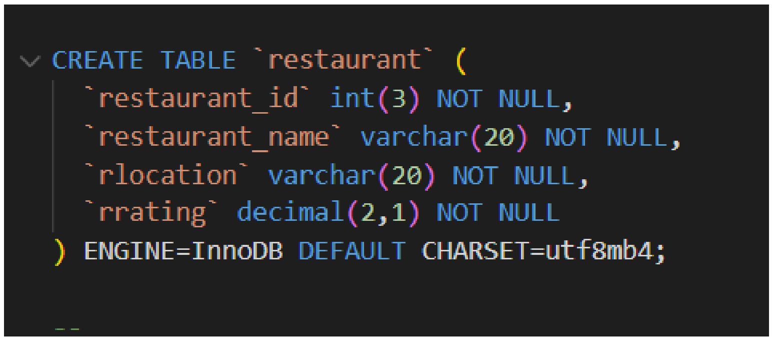
```
CREATE TABLE `order_food` (
  `order_id` int(4) NOT NULL,
  `customer_id` int(4) NOT NULL,
  `restaurant_id` int(3) NOT NULL,
  `total_cost` int(10) DEFAULT NULL,
  `order_time` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  `order_status` varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
```

5) To create table ‘order_invoice’



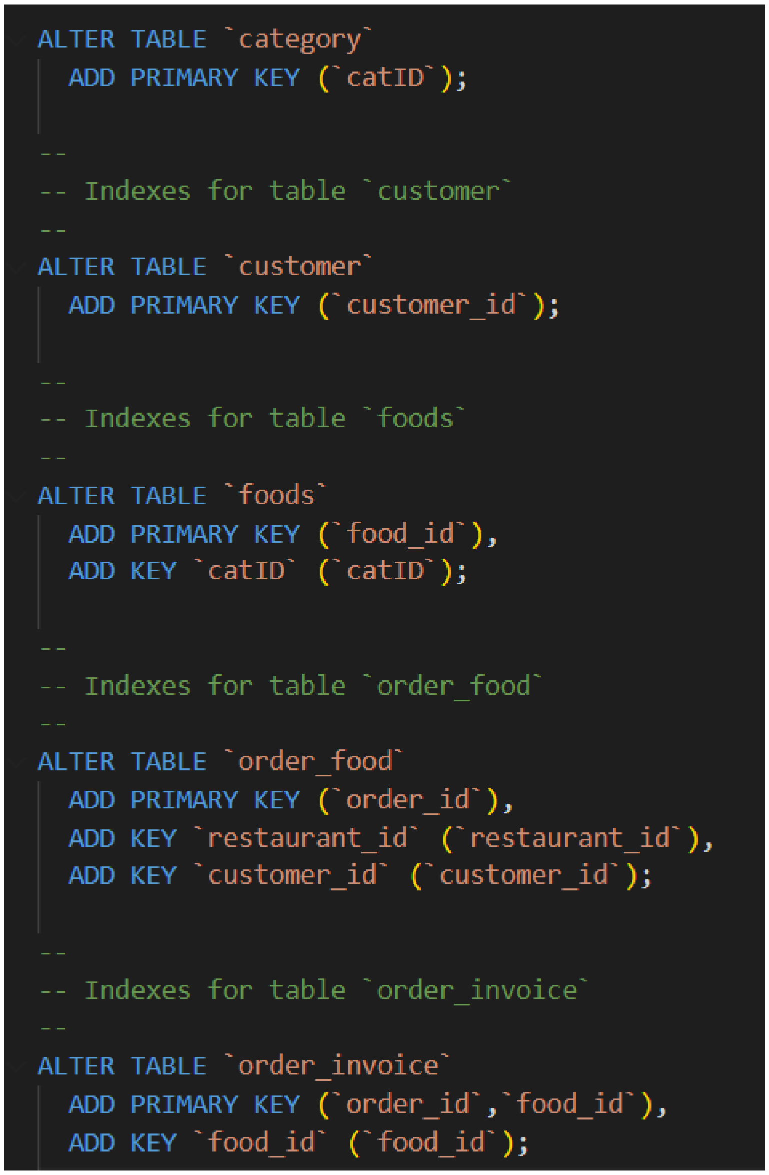
```
CREATE TABLE `order_invoice` (
  `order_id` int(11) NOT NULL,
  `food_id` int(11) NOT NULL,
  `Quantity` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

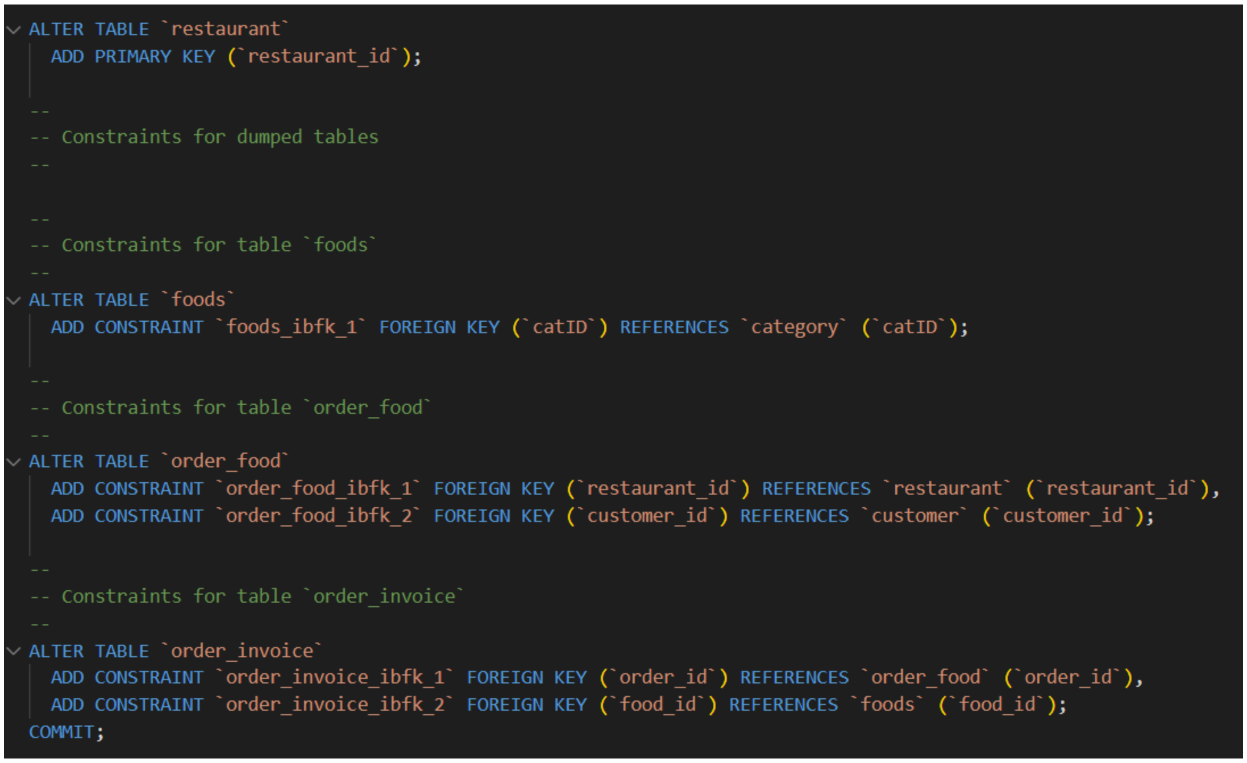
6) To create table ‘restaurant’



```
CREATE TABLE `restaurant` (
  `restaurant_id` int(3) NOT NULL,
  `restaurant_name` varchar(20) NOT NULL,
  `rlocation` varchar(20) NOT NULL,
  `rrating` decimal(2,1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
```

7) Alter Statements

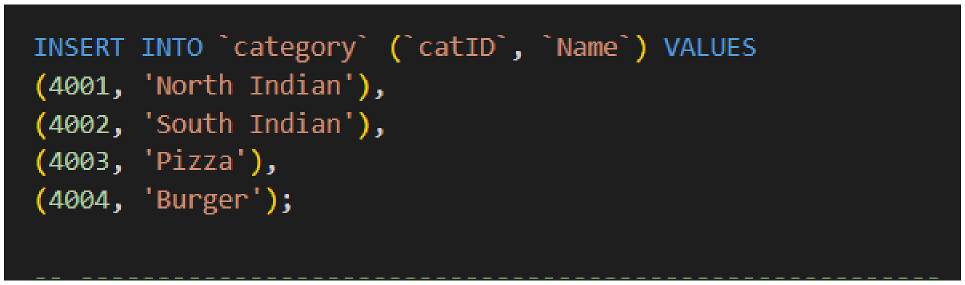

```sql
ALTER TABLE `category`
  ADD PRIMARY KEY (`catID`);

--
-- Indexes for table `customer`
--
ALTER TABLE `customer`
  ADD PRIMARY KEY (`customer_id`);

--
-- Indexes for table `foods`
--
ALTER TABLE `foods`
  ADD PRIMARY KEY (`food_id`),
  ADD KEY `catID` (`catID`);

--
-- Indexes for table `order_food`
--
ALTER TABLE `order_food`
  ADD PRIMARY KEY (`order_id`),
  ADD KEY `restaurant_id` (`restaurant_id`),
  ADD KEY `customer_id` (`customer_id`);

--
-- Indexes for table `order_invoice`
--
ALTER TABLE `order_invoice`
  ADD PRIMARY KEY (`order_id`,`food_id`),
  ADD KEY `food_id` (`food_id`);
```



```sql
ALTER TABLE `restaurant`
  ADD PRIMARY KEY (`restaurant_id`);

--
-- Constraints for dumped tables
--

--
-- Constraints for table `foods`
--
ALTER TABLE `foods`
  ADD CONSTRAINT `foods_ibfk_1` FOREIGN KEY (`catID`) REFERENCES `category` (`catID`);

--
-- Constraints for table `order_food`
--
ALTER TABLE `order_food`
  ADD CONSTRAINT `order_food_ibfk_1` FOREIGN KEY (`restaurant_id`) REFERENCES `restaurant` (`restaurant_id`),
  ADD CONSTRAINT `order_food_ibfk_2` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`customer_id`);

--
-- Constraints for table `order_invoice`
--
ALTER TABLE `order_invoice`
  ADD CONSTRAINT `order_invoice_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES `order_food` (`order_id`),
  ADD CONSTRAINT `order_invoice_ibfk_2` FOREIGN KEY (`food_id`) REFERENCES `foods` (`food_id`);
COMMIT;
```

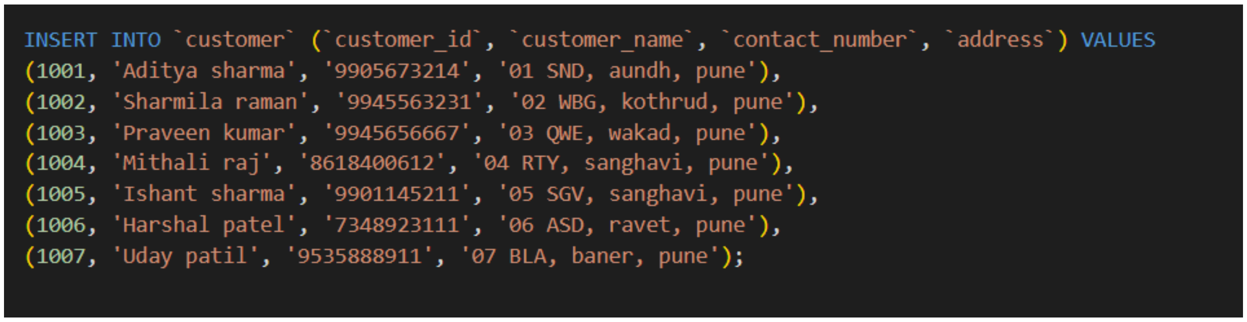
5. Populating the Database

1) Inserting into category table



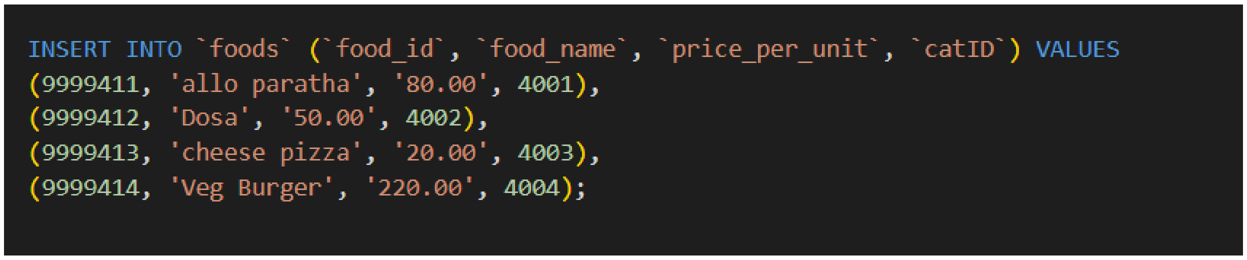
```sql
INSERT INTO `category` (`catID`, `Name`) VALUES
(4001, 'North Indian'),
(4002, 'South Indian'),
(4003, 'Pizza'),
(4004, 'Burger');

-- ------------------------------------------------------------
```

2) Inserting into customer table



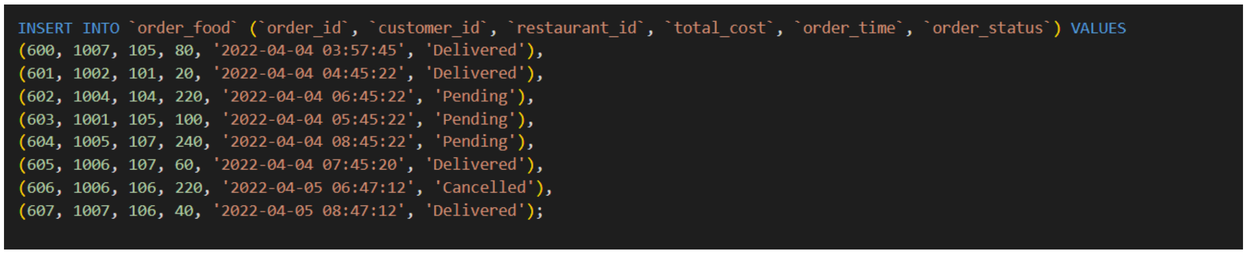
```sql
INSERT INTO `customer` (`customer_id`, `customer_name`, `contact_number`, `address`) VALUES
(1001, 'Aditya sharma', '9905673214', '01 SND, aundh, pune'),
(1002, 'Sharmila raman', '9945563231', '02 WBG, kothrud, pune'),
(1003, 'Praveen kumar', '9945656667', '03 QWE, wakad, pune'),
(1004, 'Mithali raj', '8618400612', '04 RTY, sanghavi, pune'),
(1005, 'Ishant sharma', '9901145211', '05 SGV, sanghavi, pune'),
(1006, 'Harshal patel', '7348923111', '06 ASD, ravet, pune'),
(1007, 'Uday patil', '9535888911', '07 BLA, baner, pune');
```

3) Inserting into foods table



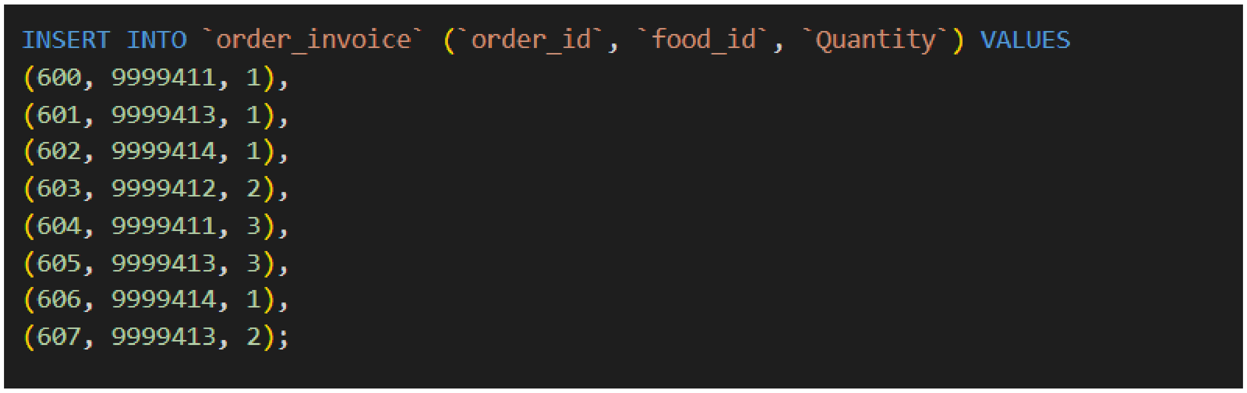
```sql
INSERT INTO `foods` (`food_id`, `food_name`, `price_per_unit`, `catID`) VALUES
(9999411, 'allo paratha', '80.00', 4001),
(9999412, 'Dosa', '50.00', 4002),
(9999413, 'cheese pizza', '20.00', 4003),
(9999414, 'Veg Burger', '220.00', 4004);
```

4) Inserting into order_food table



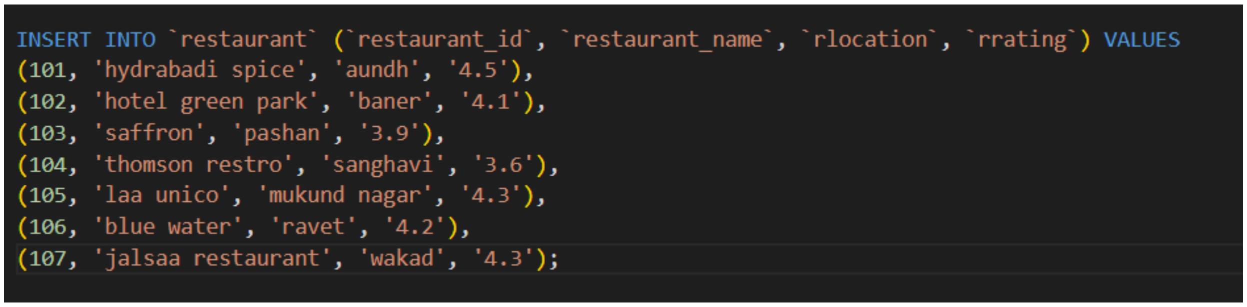
```sql
INSERT INTO `order_food` (`order_id`, `customer_id`, `restaurant_id`, `total_cost`, `order_time`, `order_status`) VALUES
(600, 1007, 105, 80, '2022-04-04 03:57:45', 'Delivered'),
(601, 1002, 101, 20, '2022-04-04 04:45:22', 'Delivered'),
(602, 1004, 104, 220, '2022-04-04 06:45:22', 'Pending'),
(603, 1001, 105, 100, '2022-04-04 05:45:22', 'Pending'),
(604, 1005, 107, 240, '2022-04-04 08:45:22', 'Pending'),
(605, 1006, 107, 60, '2022-04-04 07:45:20', 'Delivered'),
(606, 1006, 106, 220, '2022-04-05 06:47:12', 'Cancelled'),
(607, 1007, 106, 40, '2022-04-05 08:47:12', 'Delivered');
```

5) **Inserting into order_invoice table**



```sql
INSERT INTO `order_invoice` (`order_id`, `food_id`, `Quantity`) VALUES
(600, 9999411, 1),
(601, 9999413, 1),
(602, 9999414, 1),
(603, 9999412, 2),
(604, 9999411, 3),
(605, 9999413, 3),
(606, 9999414, 1),
(607, 9999413, 2);
```

6) **Inserting into restaurant table**

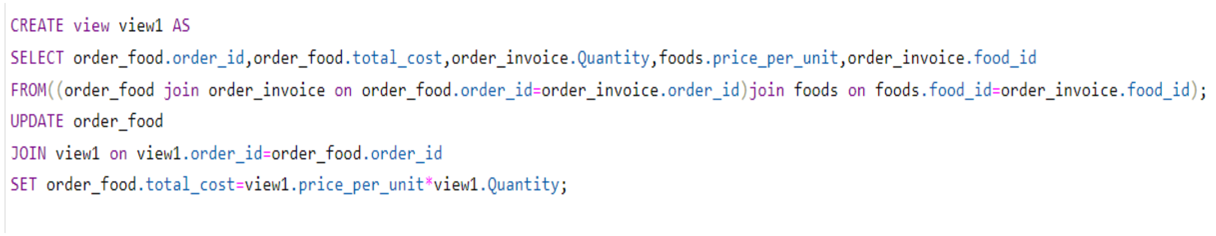


```sql
INSERT INTO `restaurant` (`restaurant_id`, `restaurant_name`, `rlocation`, `rrating`) VALUES
(101, 'hydrabadi spice', 'aundh', '4.5'),
(102, 'hotel green park', 'baner', '4.1'),
(103, 'saffron', 'pashan', '3.9'),
(104, 'thomson restro', 'sanghavi', '3.6'),
(105, 'laa unico', 'mukund nagar', '4.3'),
(106, 'blue water', 'ravet', '4.2'),
(107, 'jalsaa restaurant', 'wakad', '4.3');
```

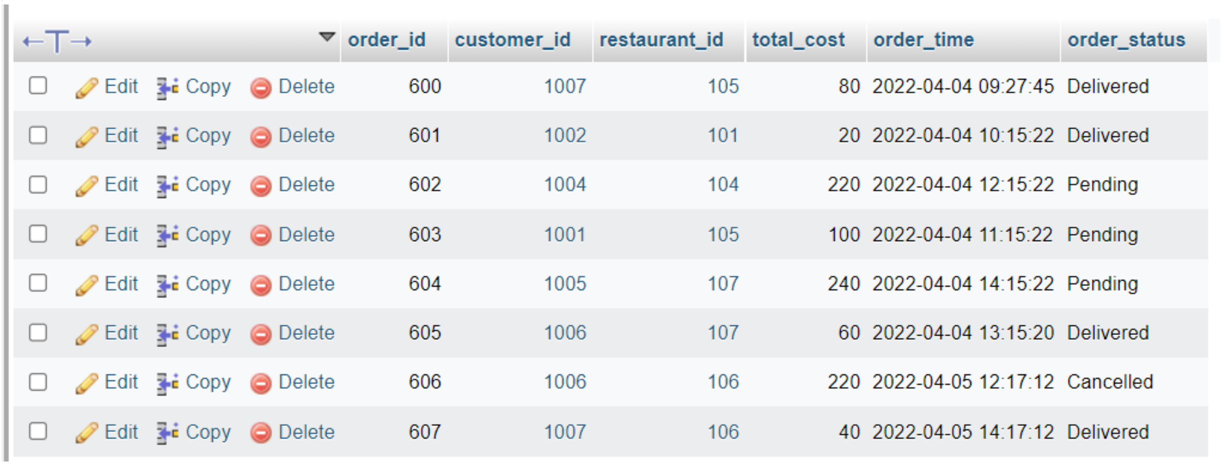
6. Join Queries

Showcase at least 4 join queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

1) Update the total cost of each order using view table.

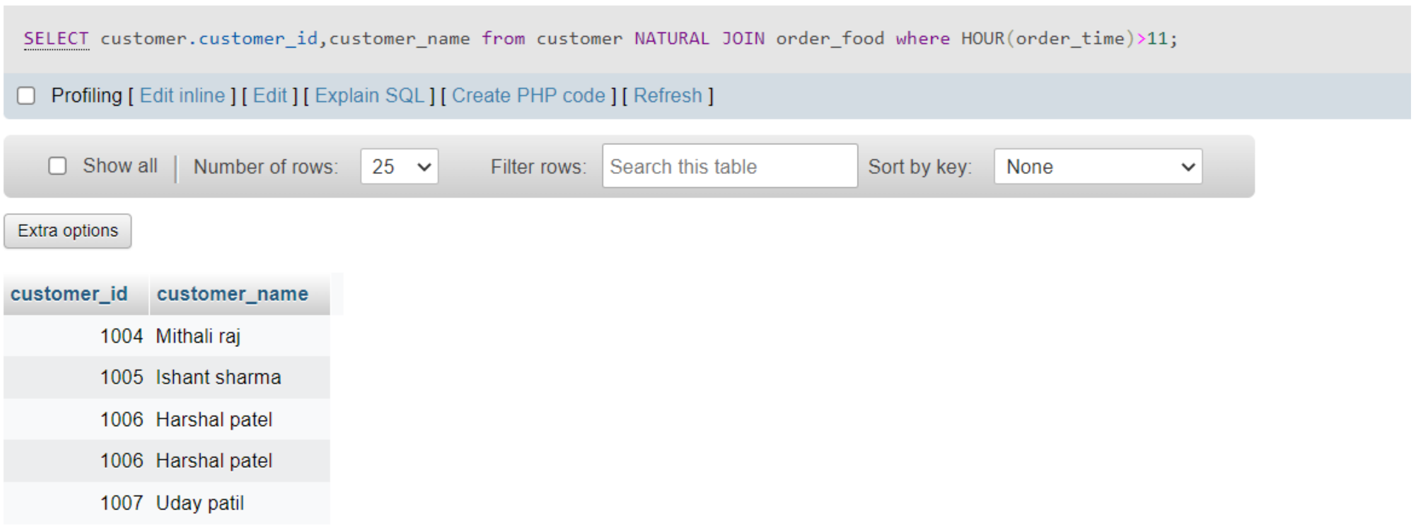


```
CREATE view view1 AS
SELECT order_food.order_id,order_food.total_cost,order_invoice.Quantity,foods.price_per_unit,order_invoice.food_id
FROM((order_food join order_invoice on order_food.order_id=order_invoice.order_id)join foods on foods.food_id=order_invoice.food_id);
UPDATE order_food
JOIN view1 on view1.order_id=order_food.order_id
SET order_food.total_cost=view1.price_per_unit*view1.Quantity;
```



←T→	order_id	customer_id	restaurant_id	total_cost	order_time	order_status
☐ Edit Copy Delete	600	1007	105	80	2022-04-04 09:27:45	Delivered
☐ Edit Copy Delete	601	1002	101	20	2022-04-04 10:15:22	Delivered
☐ Edit Copy Delete	602	1004	104	220	2022-04-04 12:15:22	Pending
☐ Edit Copy Delete	603	1001	105	100	2022-04-04 11:15:22	Pending
☐ Edit Copy Delete	604	1005	107	240	2022-04-04 14:15:22	Pending
☐ Edit Copy Delete	605	1006	107	60	2022-04-04 13:15:20	Delivered
☐ Edit Copy Delete	606	1006	106	220	2022-04-05 12:17:12	Cancelled
☐ Edit Copy Delete	607	1007	106	40	2022-04-05 14:17:12	Delivered

2) Find list of customers who have ordered after 11 am.

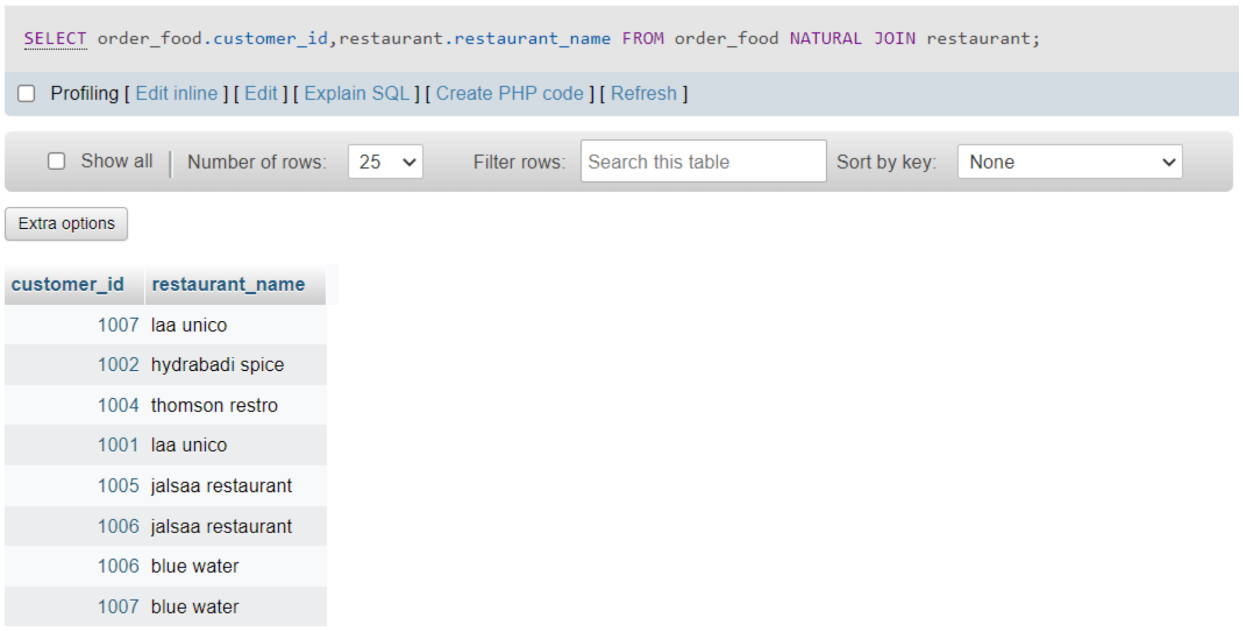


```
SELECT customer.customer_id,customer_name from customer NATURAL JOIN order_food where HOUR(order_time)>11;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
☐ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

customer_id	customer_name
1004	Mithali raj
1005	Ishant sharma
1006	Harshal patel
1006	Harshal patel
1007	Uday patil

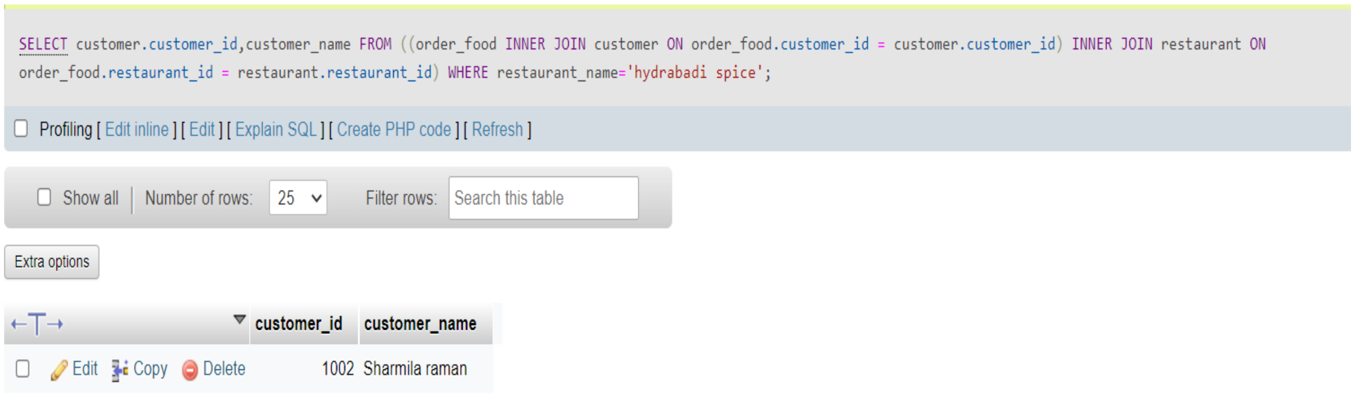
3) Find list of customer ids and the restaurant names they have ordered from.



```
SELECT order_food.customer_id,restaurant.restaurant_name FROM order_food NATURAL JOIN restaurant;
```

customer_id	restaurant_name
1007	laa unico
1002	hydrabadi spice
1004	thomson restro
1001	laa unico
1005	jalsaa restaurant
1006	jalsaa restaurant
1006	blue water
1007	blue water

4) Find the list of customers who have ordered from hydrabadi spice.



```
SELECT customer.customer_id,customer_name FROM ((order_food INNER JOIN customer ON order_food.customer_id = customer.customer_id) INNER JOIN restaurant ON
order_food.restaurant_id = restaurant.restaurant_id) WHERE restaurant_name='hydrabadi spice';
```

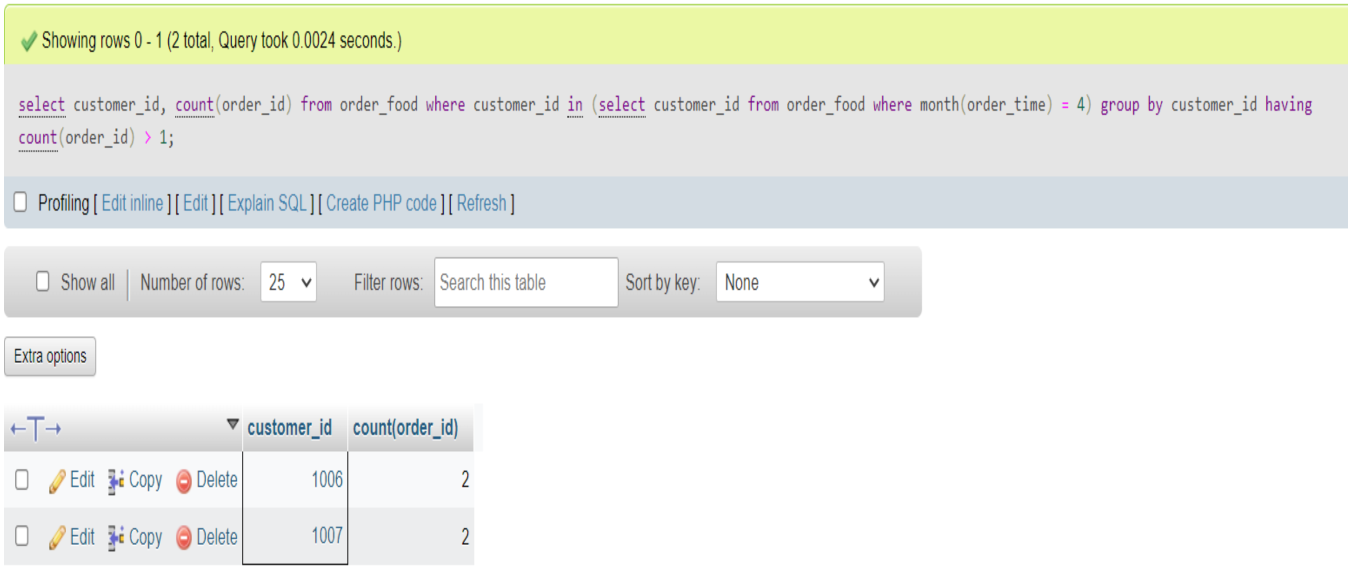
customer_id	customer_name
1002	Sharmila raman

7. Aggregate Functions

Showcase at least 4 Aggregate function queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

1) **Find the list of customers who have made more than 1 order in a particular month.**

.



✓ Showing rows 0 - 1 (2 total, Query took 0.0024 seconds.)

```
select customer_id, count(order_id) from order_food where customer_id in (select customer_id from order_food where month(order_time) = 4) group by customer_id having count(order_id) > 1;
```

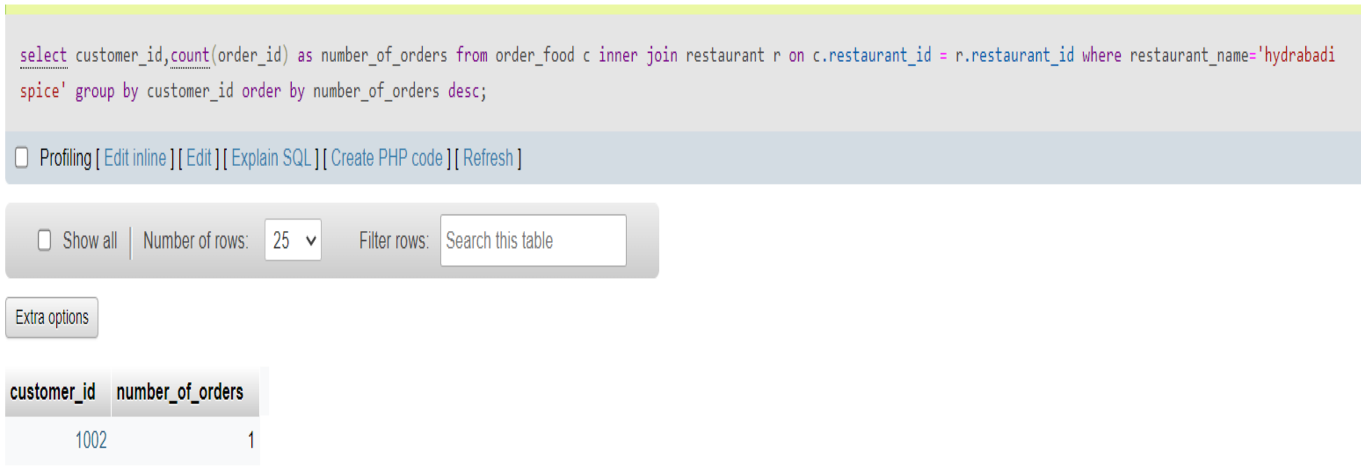
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	customer_id	count(order_id)
Edit Copy Delete	1006	2
Edit Copy Delete	1007	2

2) **Find the list of customers and their total number of orders from hydrabadi spice.**



```
select customer_id,count(order_id) as number_of_orders from order_food c inner join restaurant r on c.restaurant_id = r.restaurant_id where restaurant_name='hydrabadi spice' group by customer_id order by number_of_orders desc;
```

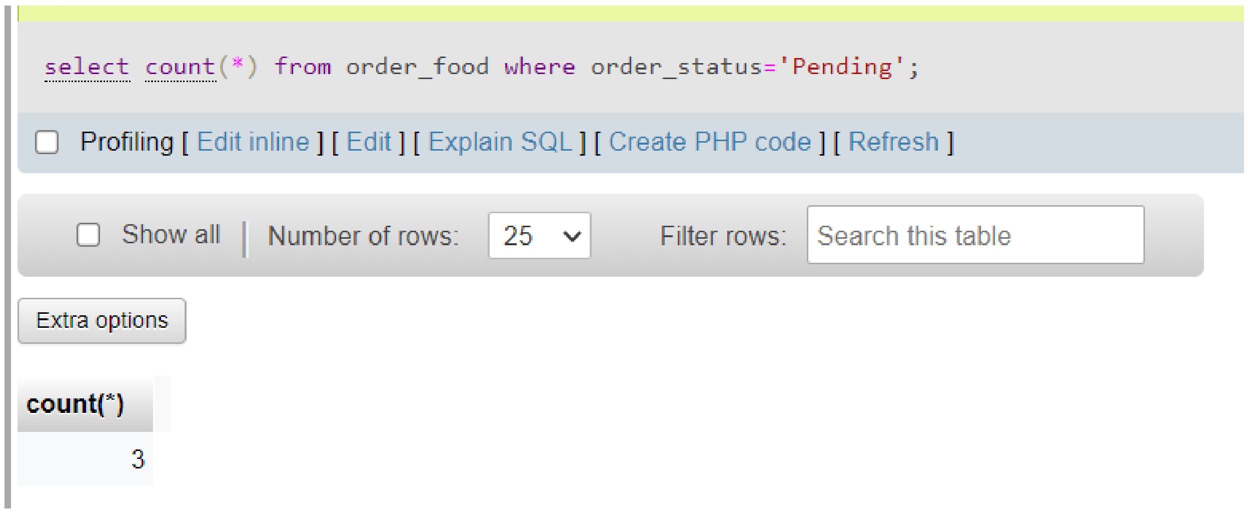
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

customer_id	number_of_orders
1002	1

3) **Find the number of orders whose order status is still pending.**

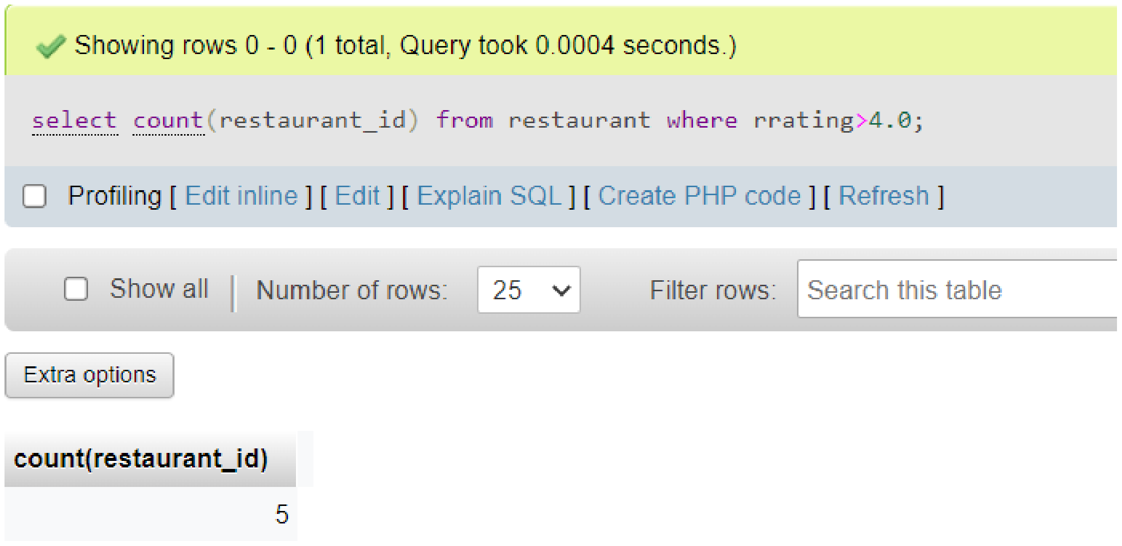

```
select count(*) from order_food where order_status='Pending';
```
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

count(*)
3

4) Find the number of restaurants whose rating is more than 4.0.


Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
select count(restaurant_id) from restaurant where rrating>4.0;
```
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

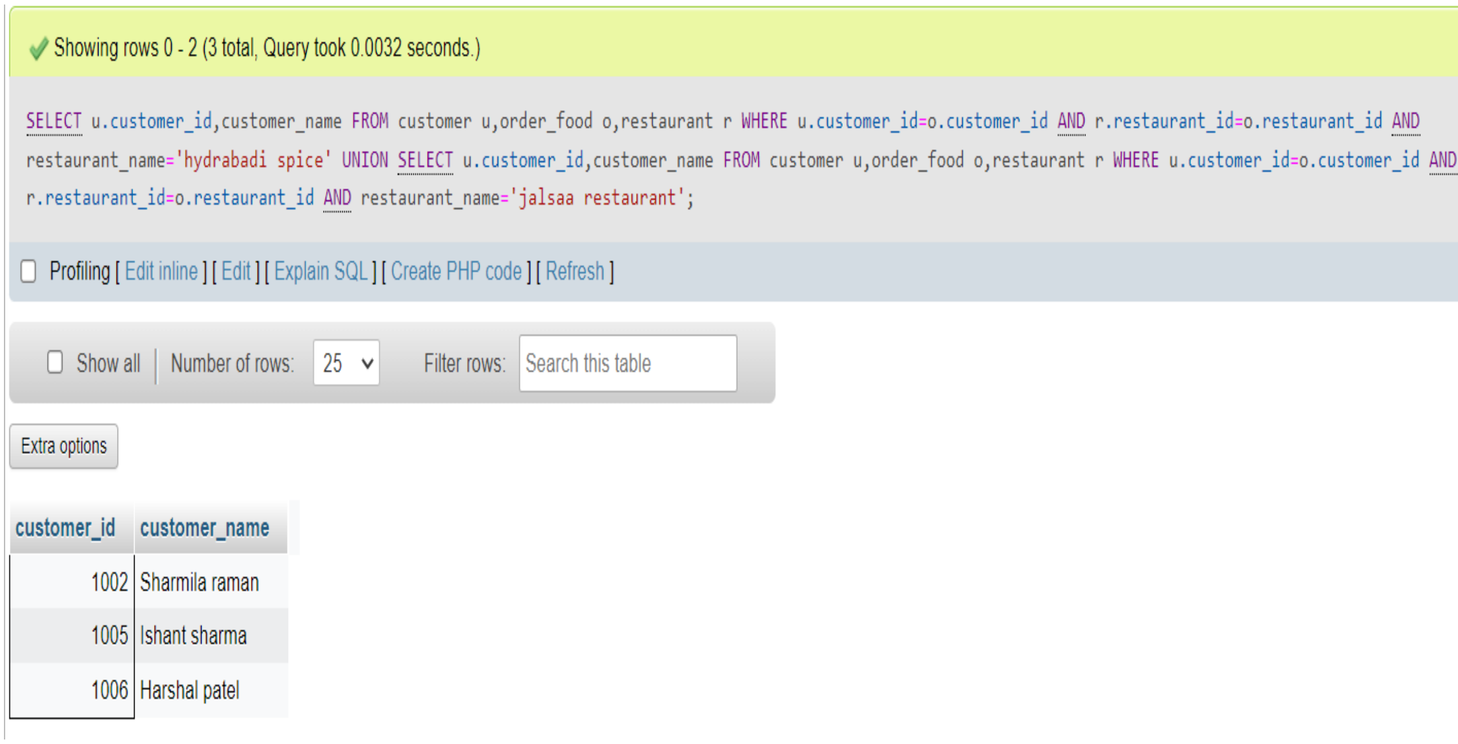
Extra options

count(restaurant_id)
5

8. Set Operations

Showcase at least 4 Set Operations queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

1) Find the list of customers names and ids who have ordered from restaurants hyderabadi spice and jalsaa restaurant.


✓ Showing rows 0 - 2 (3 total, Query took 0.0032 seconds.)

```
SELECT u.customer_id,customer_name FROM customer u,order_food o,restaurant r WHERE u.customer_id=o.customer_id AND r.restaurant_id=o.restaurant_id AND
restaurant_name='hydrabadi spice' UNION SELECT u.customer_id,customer_name FROM customer u,order_food o,restaurant r WHERE u.customer_id=o.customer_id AND
r.restaurant_id=o.restaurant_id AND restaurant_name='jalsaa restaurant';
```

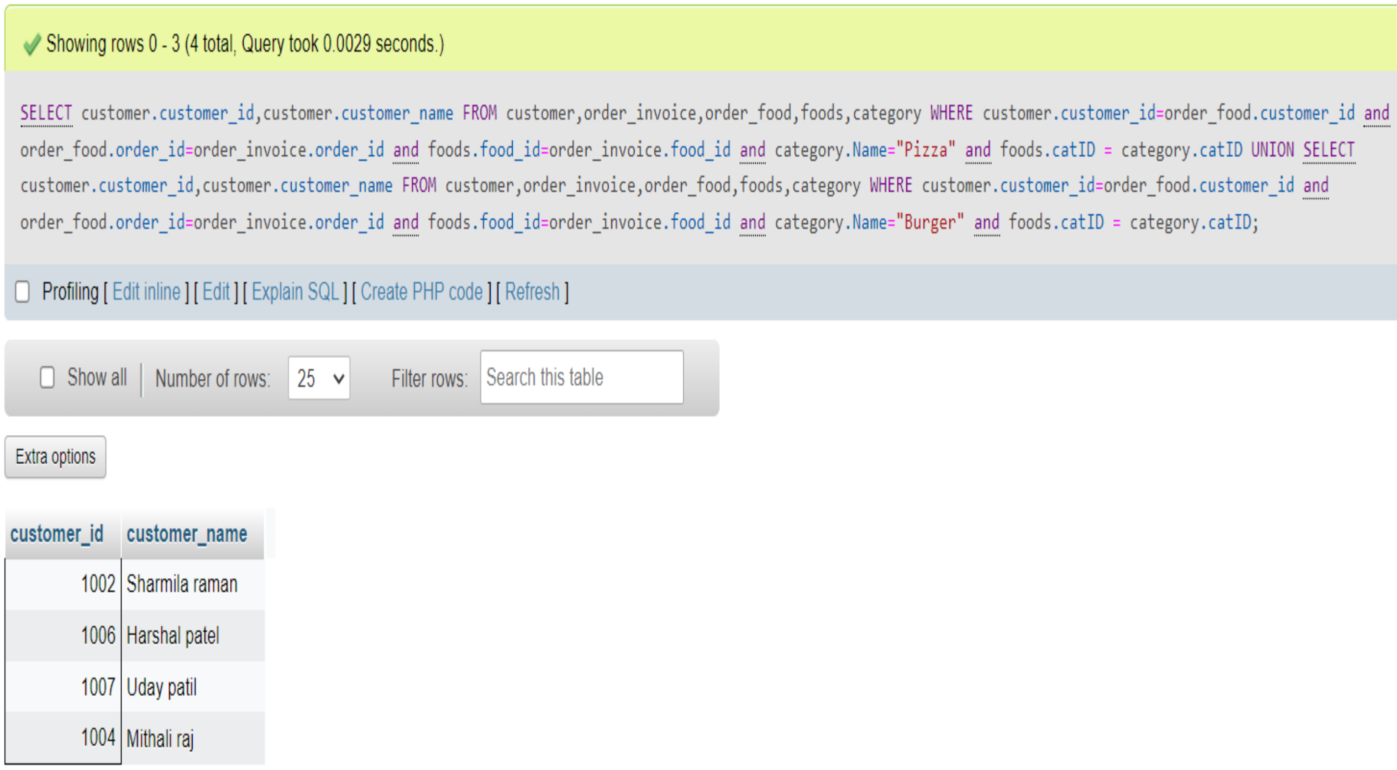
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

customer_id	customer_name
1002	Sharmila raman
1005	Ishant sharma
1006	Harshal patel

2) **Find the list of customers who have ordered either pizza category or burger category.**


✓ Showing rows 0 - 3 (4 total, Query took 0.0029 seconds.)

```
SELECT customer.customer_id,customer.customer_name FROM customer,order_invoice,order_food,foods,category WHERE customer.customer_id=order_food.customer_id and
order_food.order_id=order_invoice.order_id and foods.food_id=order_invoice.food_id and category.Name="Pizza" and foods.catID = category.catID UNION SELECT
customer.customer_id,customer.customer_name FROM customer,order_invoice,order_food,foods,category WHERE customer.customer_id=order_food.customer_id and
order_food.order_id=order_invoice.order_id and foods.food_id=order_invoice.food_id and category.Name="Burger" and foods.catID = category.catID;
```

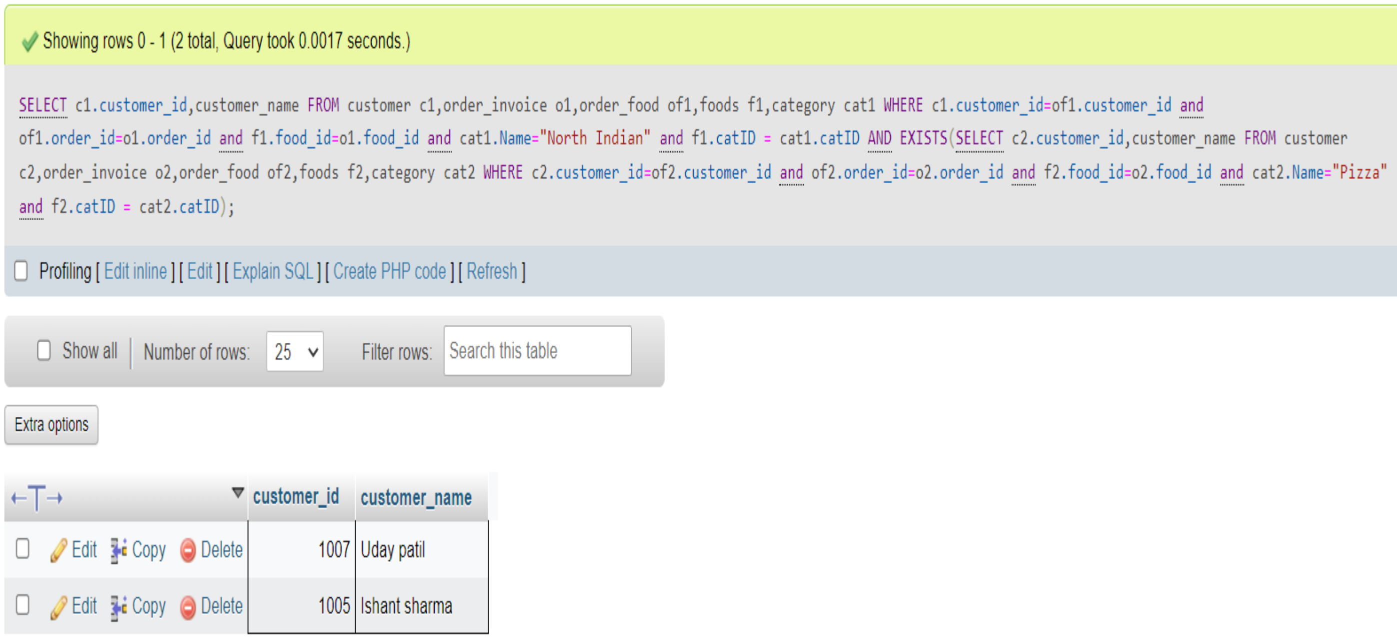
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

customer_id	customer_name
1002	Sharmila raman
1006	Harshal patel
1007	Uday patil
1004	Mithali raj

3) **Find the list of customers who have ordered north Indian category and Pizza category.**

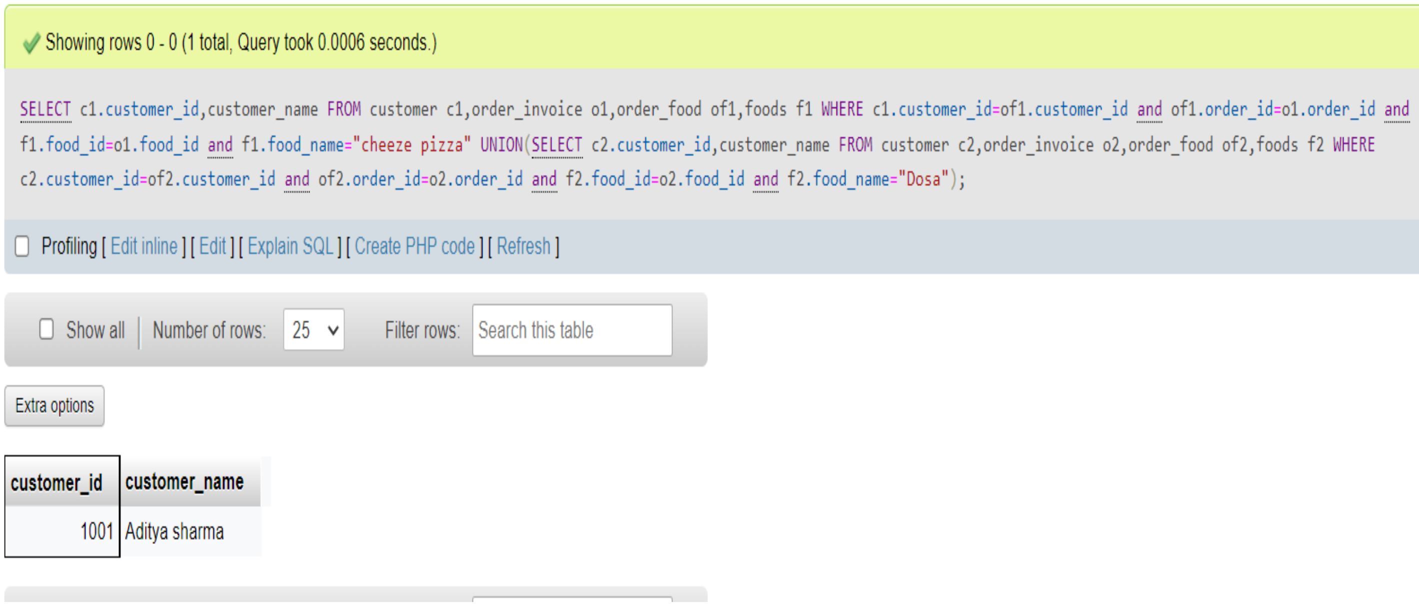


Showing rows 0 - 1 (2 total, Query took 0.0017 seconds.)

```
SELECT c1.customer_id,customer_name FROM customer c1,order_invoice o1,order_food of1,foods f1,category cat1 WHERE c1.customer_id=of1.customer_id and of1.order_id=o1.order_id and f1.food_id=o1.food_id and cat1.Name="North Indian" and f1.catID = cat1.catID AND EXISTS(SELECT c2.customer_id,customer_name FROM customer c2,order_invoice o2,order_food of2,foods f2,category cat2 WHERE c2.customer_id=of2.customer_id and of2.order_id=o2.order_id and f2.food_id=o2.food_id and cat2.Name="Pizza" and f2.catID = cat2.catID);
```

	customer_id	customer_name
Edit Copy Delete	1007	Uday patil
Edit Copy Delete	1005	Ishant sharma

4) **Find the list of customers who have ordered cheese pizza and dosa.**



Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)

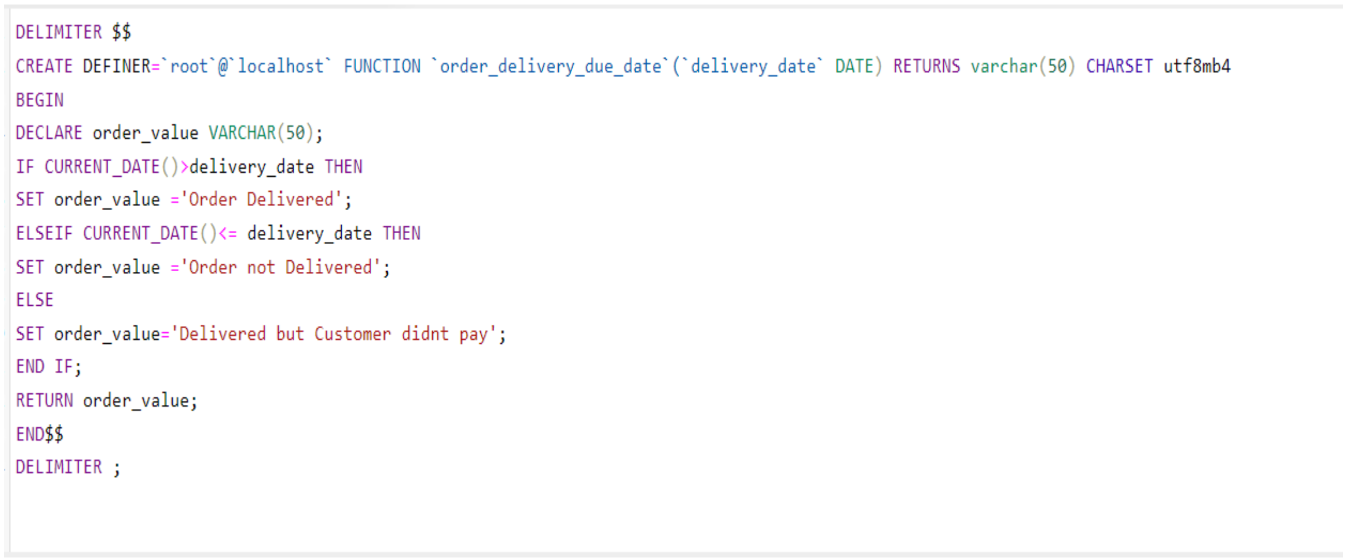
```
SELECT c1.customer_id,customer_name FROM customer c1,order_invoice o1,order_food of1,foods f1 WHERE c1.customer_id=of1.customer_id and of1.order_id=o1.order_id and f1.food_id=o1.food_id and f1.food_name="cheeze pizza" UNION(SELECT c2.customer_id,customer_name FROM customer c2,order_invoice o2,order_food of2,foods f2 WHERE c2.customer_id=of2.customer_id and of2.order_id=o2.order_id and f2.food_id=o2.food_id and f2.food_name="Dosa");
```

customer_id	customer_name
1001	Aditya sharma

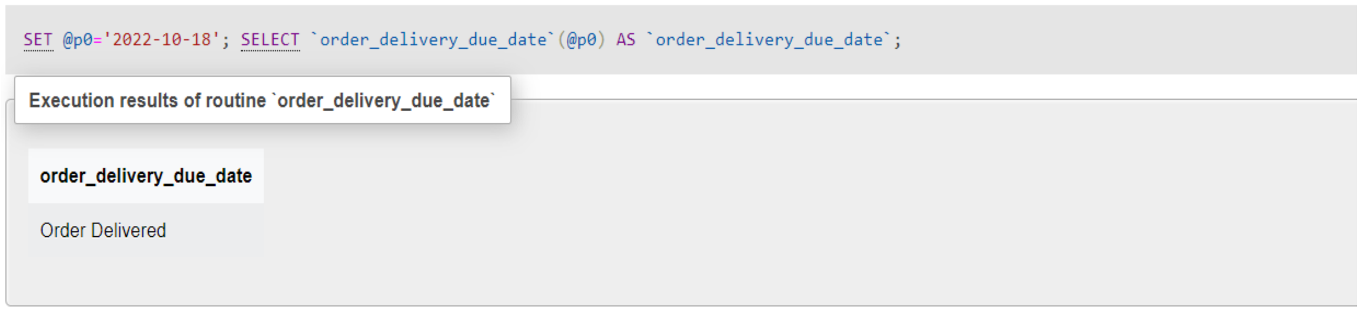
9. Functions and Procedures

Create a Function and Procedure. State the objective of the function / Procedure. Run and display the results.

1) Write a function to check if the delivery of the order is past the delivery date or not.



```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `order_delivery_due_date`(`delivery_date` DATE) RETURNS varchar(50) CHARSET utf8mb4
BEGIN
DECLARE order_value VARCHAR(50);
IF CURRENT_DATE()>delivery_date THEN
SET order_value ='Order Delivered';
ELSEIF CURRENT_DATE()<= delivery_date THEN
SET order_value ='Order not Delivered';
ELSE
SET order_value='Delivered but Customer didnt pay';
END IF;
RETURN order_value;
END$$
DELIMITER ;
```

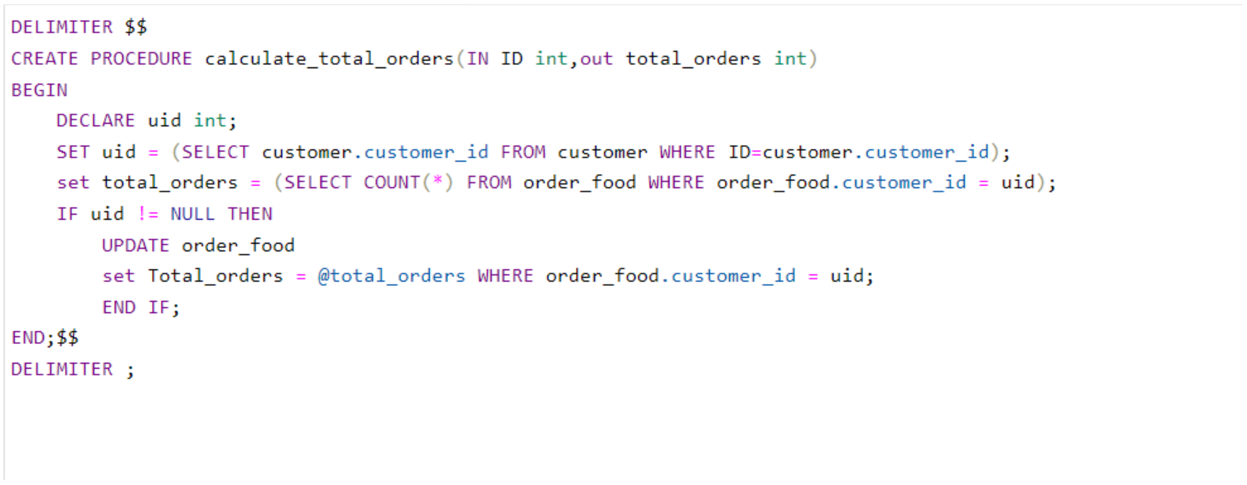


```
SET @p0='2022-10-18'; SELECT `order_delivery_due_date`(@p0) AS `order_delivery_due_date`;
```

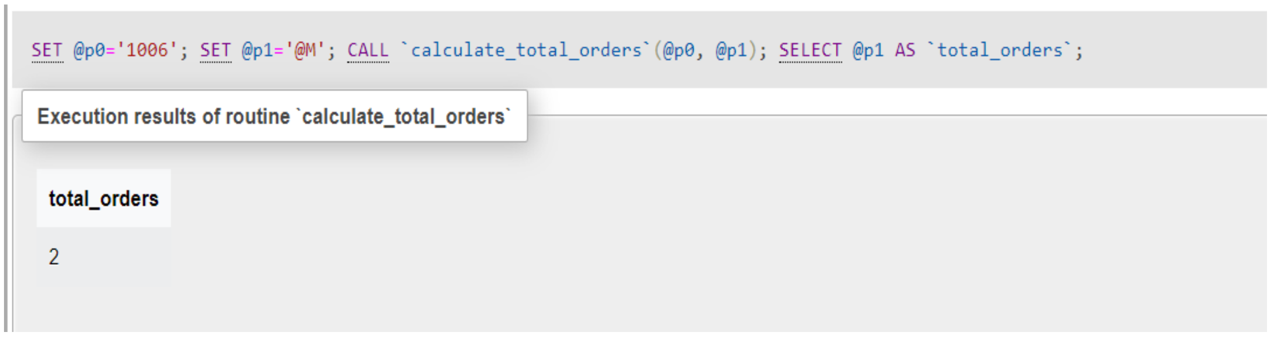
Execution results of routine `order_delivery_due_date`

```
order_delivery_due_date
Order Delivered
```

2) Write a procedure to find the total number of orders of a customer.



```
DELIMITER $$
CREATE PROCEDURE calculate_total_orders(IN ID int,out total_orders int)
BEGIN
    DECLARE uid int;
    SET uid = (SELECT customer.customer_id FROM customer WHERE ID=customer.customer_id);
    set total_orders = (SELECT COUNT(*) FROM order_food WHERE order_food.customer_id = uid);
    IF uid != NULL THEN
        UPDATE order_food
        set Total_orders = @total_orders WHERE order_food.customer_id = uid;
        END IF;
END;$$
DELIMITER ;
```



```
SET @p0='1006'; SET @p1='@M'; CALL `calculate_total_orders`(@p0, @p1); SELECT @p1 AS `total_orders`;
```

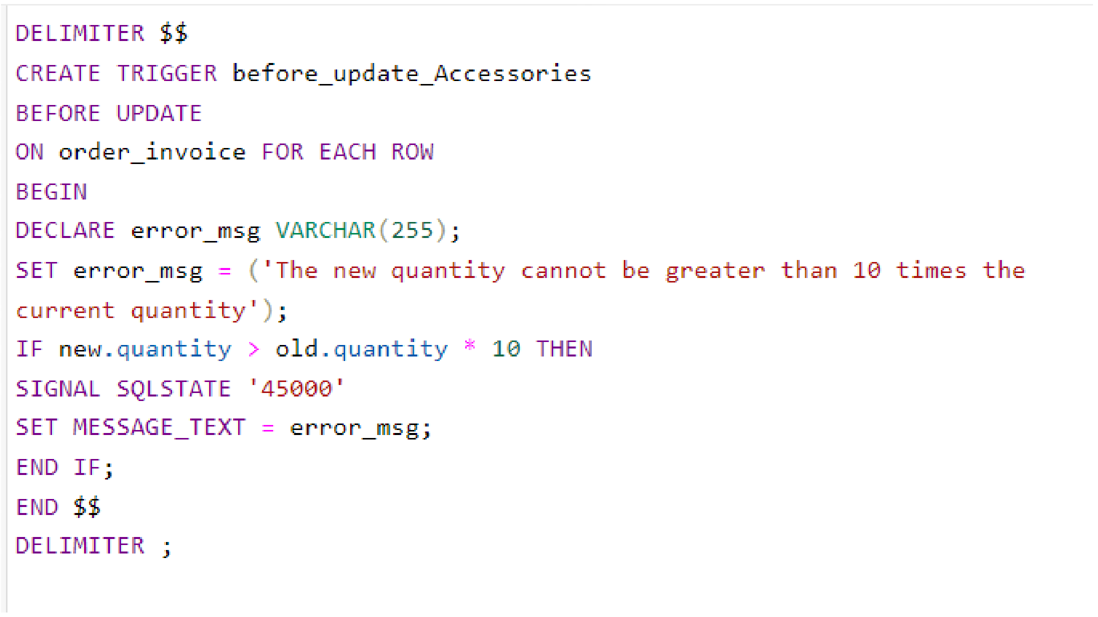
Execution results of routine `calculate_total_orders`

total_orders
2

10. Triggers and Cursors

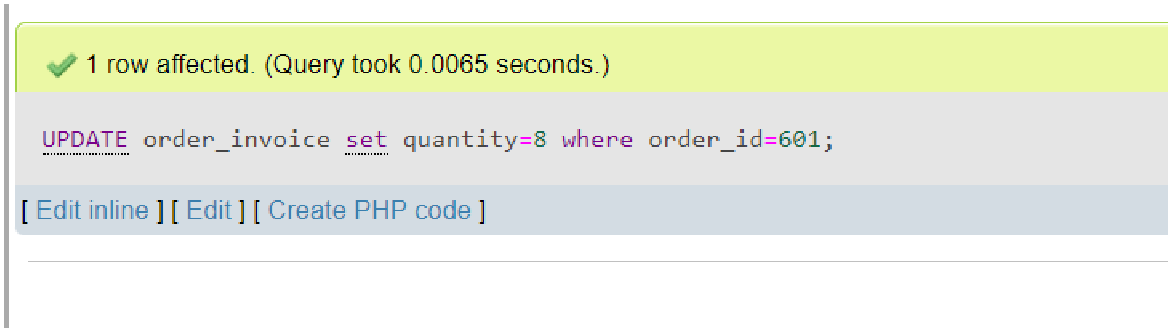
Create a Trigger and a Cursor. State the objective. Run and display the results.

1) Create a trigger to show error message and stops the updation of order_invoice table if we update the value in the quantity column to a new value ten times greater than the current value.



```
DELIMITER $$
CREATE TRIGGER before_update_Accessories
BEFORE UPDATE
ON order_invoice FOR EACH ROW
BEGIN
DECLARE error_msg VARCHAR(255);
SET error_msg = ('The new quantity cannot be greater than 10 times the
current quantity');
IF new.quantity > old.quantity * 10 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = error_msg;
END IF;
END $$
DELIMITER ;
```

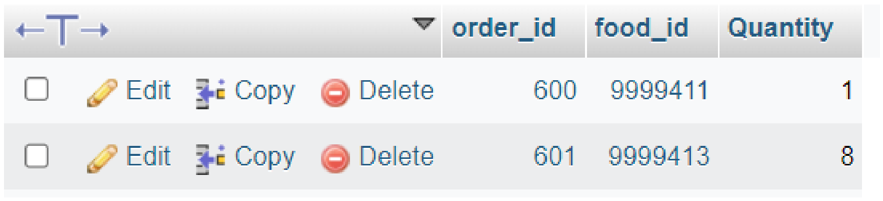
Normal case:



✓ 1 row affected. (Query took 0.0065 seconds.)

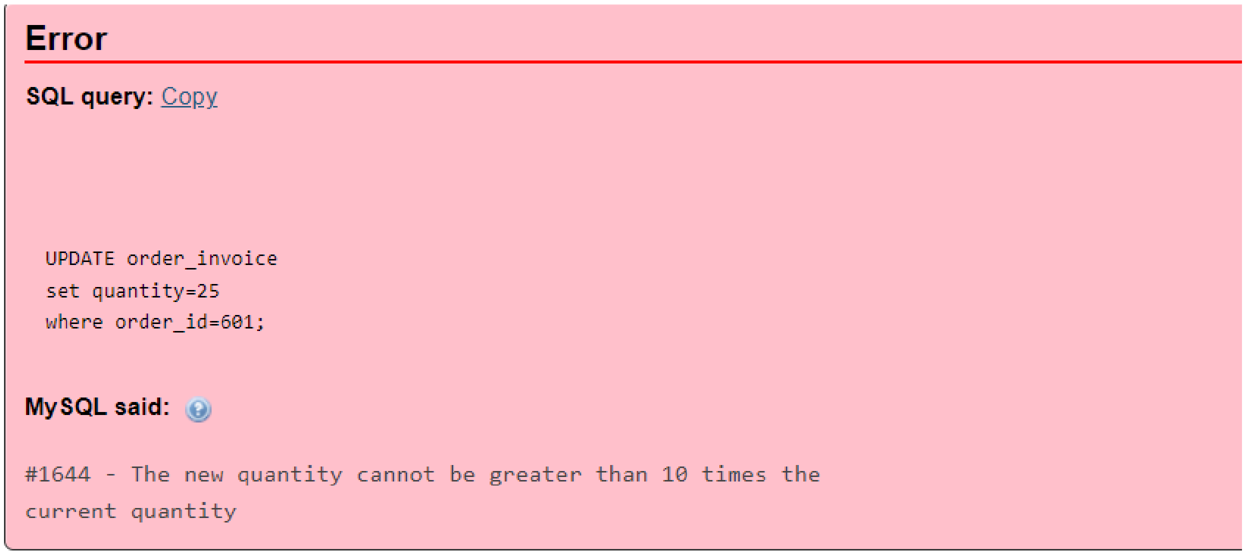
```
UPDATE order_invoice set quantity=8 where order_id=601;
```

[Edit inline] [Edit] [Create PHP code]



	order_id	food_id	Quantity
☐ Edit Copy Delete	600	9999411	1
☐ Edit Copy Delete	601	9999413	8

Error Case:



Error

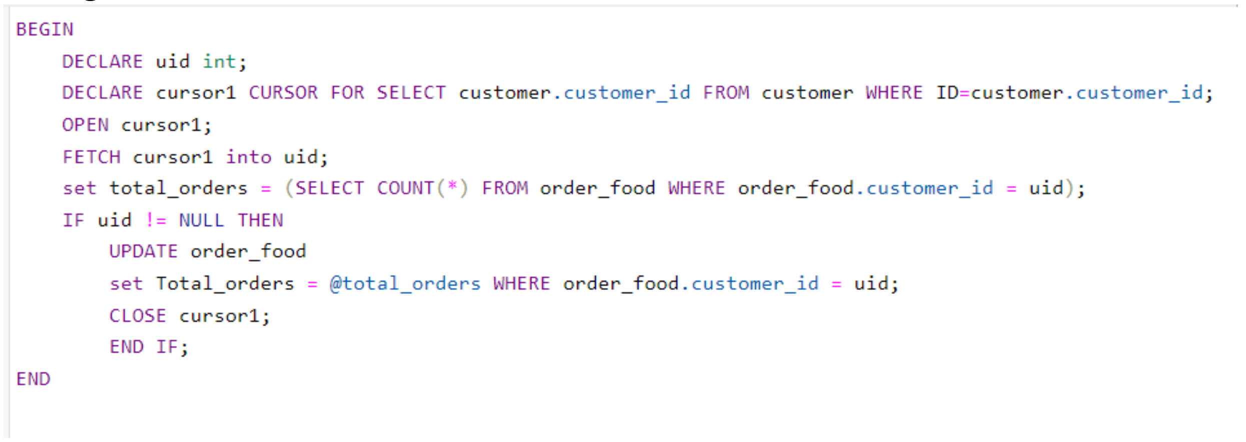
SQL query: Copy

```
UPDATE order_invoice
set quantity=25
where order_id=601;
```

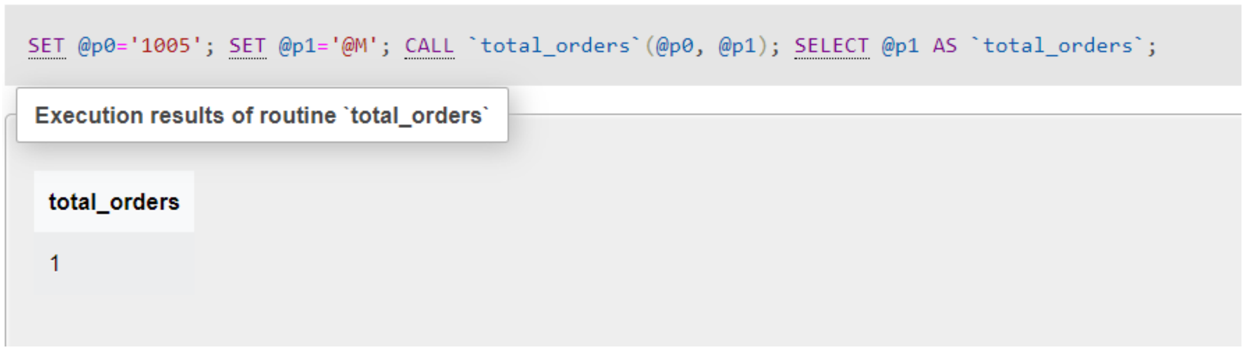
MySQL said:

```
#1644 - The new quantity cannot be greater than 10 times the
current quantity
```

2) Write a procedure to find the total number of orders of a customer using a cursor.



```
BEGIN
    DECLARE uid int;
    DECLARE cursor1 CURSOR FOR SELECT customer.customer_id FROM customer WHERE ID=customer.customer_id;
    OPEN cursor1;
    FETCH cursor1 into uid;
    set total_orders = (SELECT COUNT(*) FROM order_food WHERE order_food.customer_id = uid);
    IF uid != NULL THEN
        UPDATE order_food
        set Total_orders = @total_orders WHERE order_food.customer_id = uid;
        CLOSE cursor1;
        END IF;
END
```



```
SET @p0='1005'; SET @p1='@M'; CALL `total_orders`(@p0, @p1); SELECT @p1 AS `total_orders`;
```

Execution results of routine `total_orders`

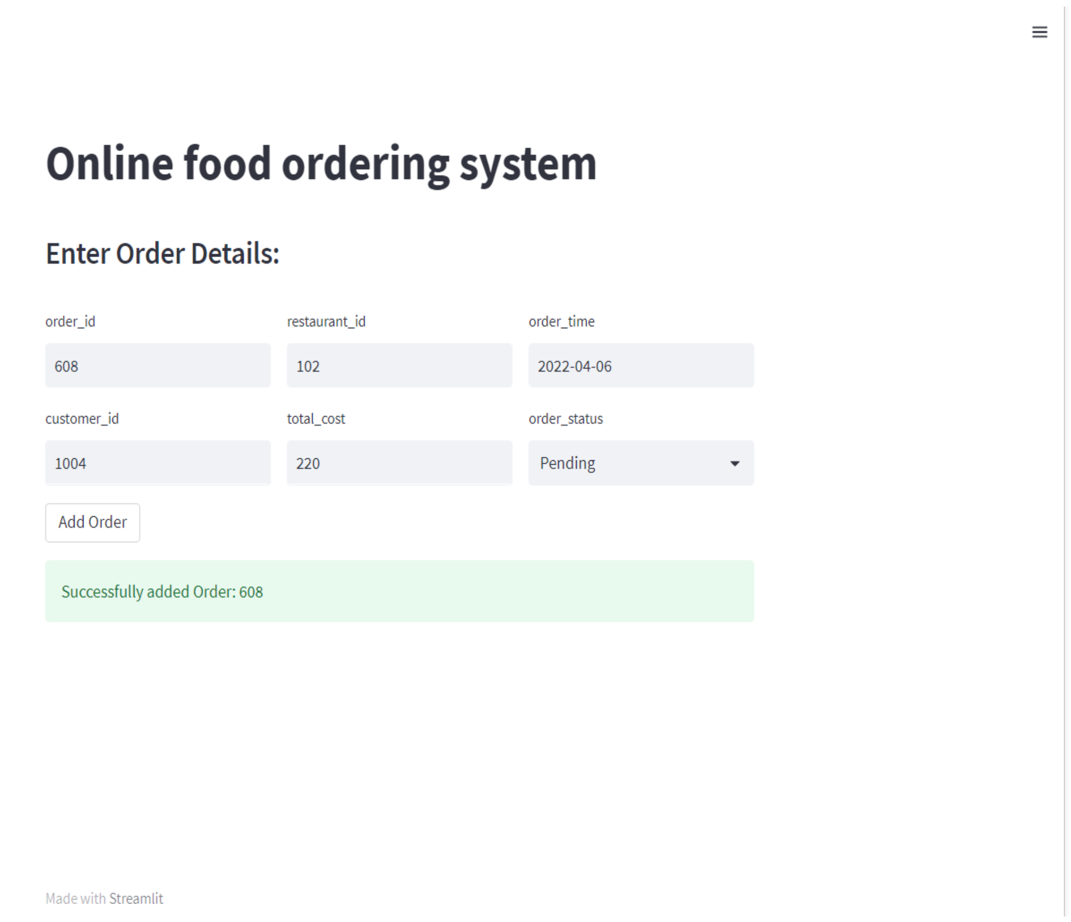
```
total_orders
1
```

11. Developing a Frontend

The frontend should support

1. Addition, Modification and Deletion of records from any chosen table
2. There should be an window to accept and run any SQL statement and display the result

1) Addition of record

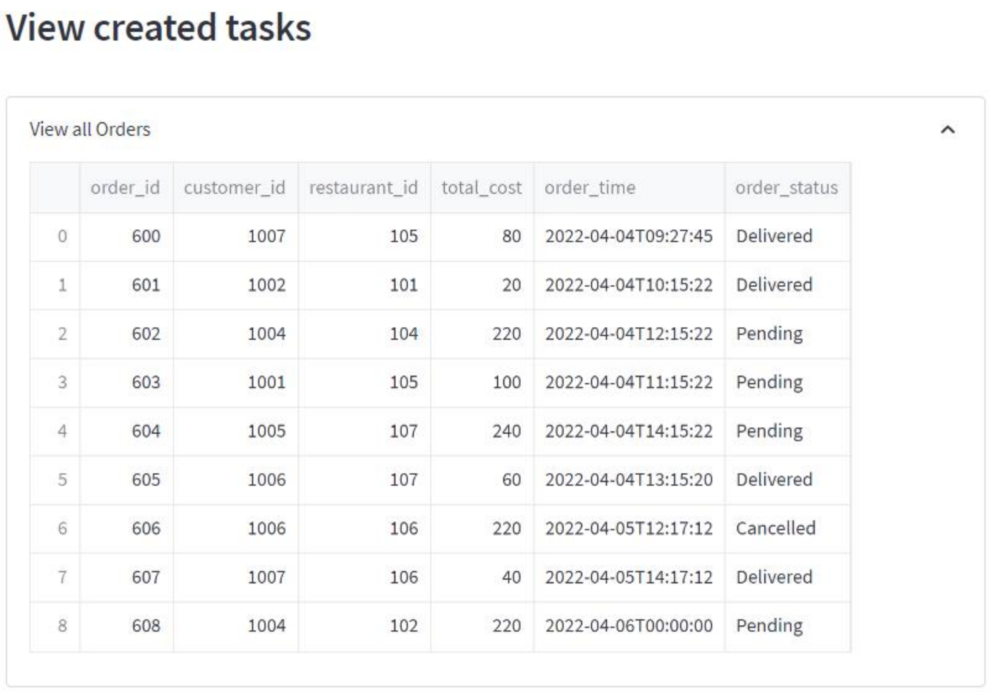


Online food ordering system

Enter Order Details:

order_id	restaurant_id	order_time
608	102	2022-04-06
customer_id	total_cost	order_status
1004	220	Pending

Successfully added Order: 608

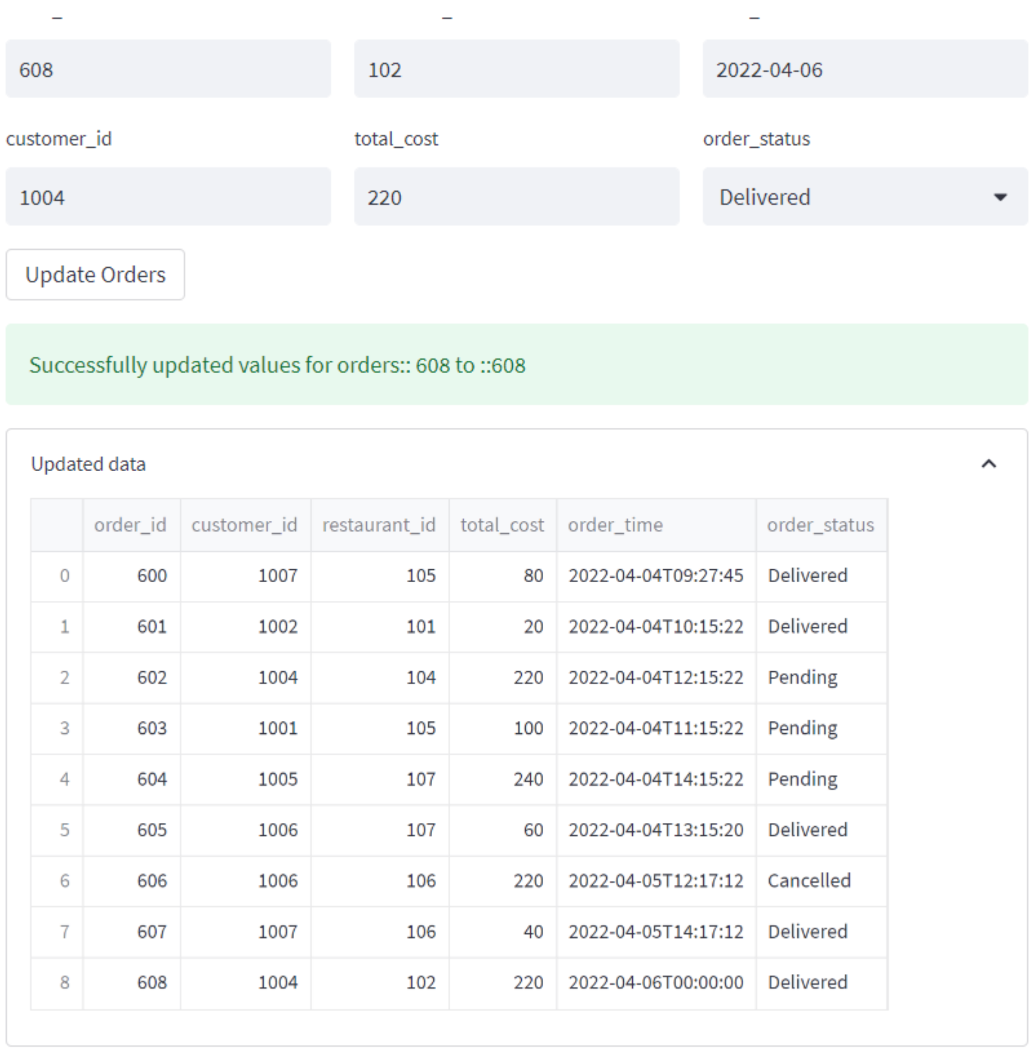


View created tasks

View all Orders

	order_id	customer_id	restaurant_id	total_cost	order_time	order_status
0	600	1007	105	80	2022-04-04T09:27:45	Delivered
1	601	1002	101	20	2022-04-04T10:15:22	Delivered
2	602	1004	104	220	2022-04-04T12:15:22	Pending
3	603	1001	105	100	2022-04-04T11:15:22	Pending
4	604	1005	107	240	2022-04-04T14:15:22	Pending
5	605	1006	107	60	2022-04-04T13:15:20	Delivered
6	606	1006	106	220	2022-04-05T12:17:12	Cancelled
7	607	1007	106	40	2022-04-05T14:17:12	Delivered
8	608	1004	102	220	2022-04-06T00:00:00	Pending

2) Modification of record



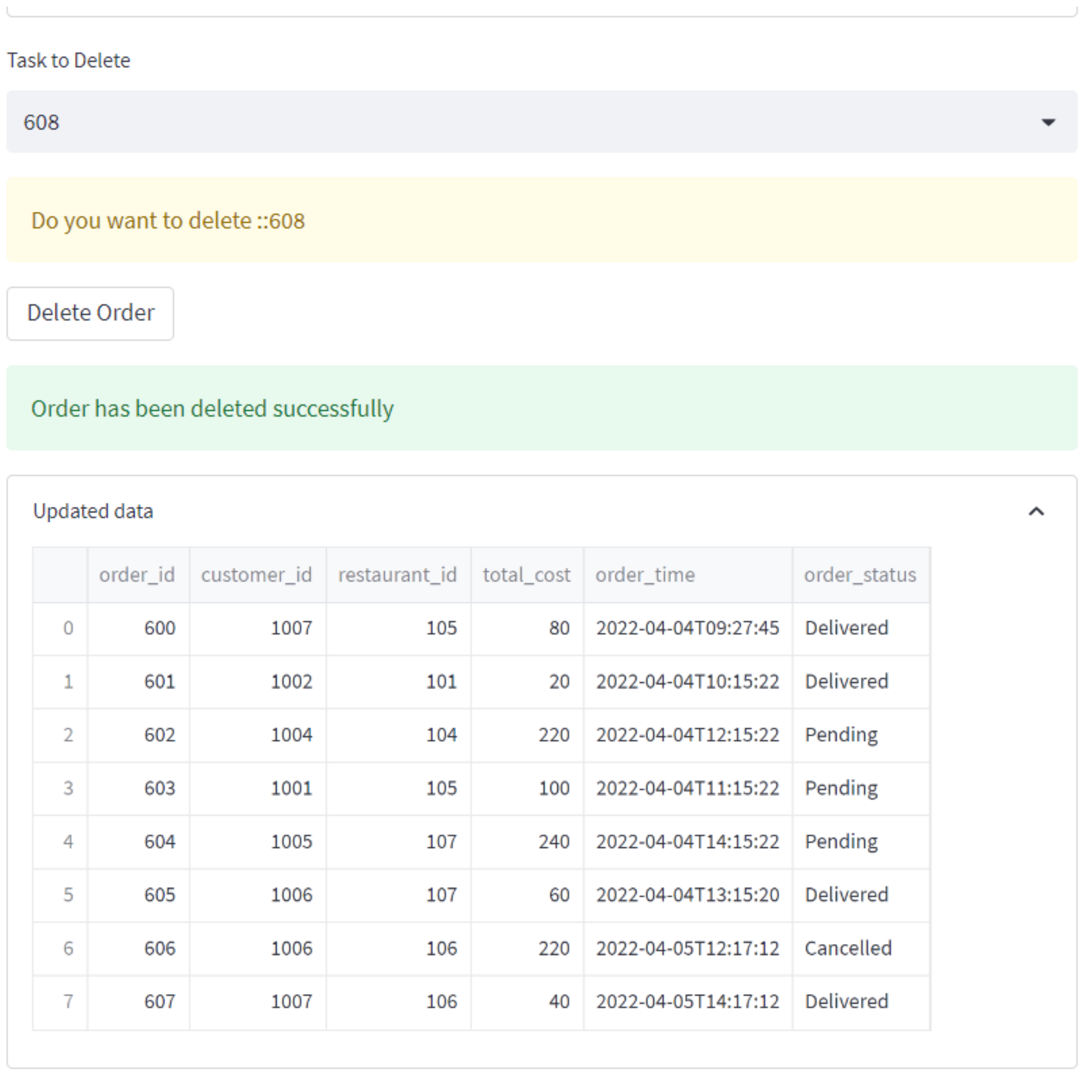
608	102	2022-04-06
customer_id	total_cost	order_status
1004	220	Delivered

Successfully updated values for orders:: 608 to ::608

Updated data

	order_id	customer_id	restaurant_id	total_cost	order_time	order_status
0	600	1007	105	80	2022-04-04T09:27:45	Delivered
1	601	1002	101	20	2022-04-04T10:15:22	Delivered
2	602	1004	104	220	2022-04-04T12:15:22	Pending
3	603	1001	105	100	2022-04-04T11:15:22	Pending
4	604	1005	107	240	2022-04-04T14:15:22	Pending
5	605	1006	107	60	2022-04-04T13:15:20	Delivered
6	606	1006	106	220	2022-04-05T12:17:12	Cancelled
7	607	1007	106	40	2022-04-05T14:17:12	Delivered
8	608	1004	102	220	2022-04-06T00:00:00	Delivered

3) Deletion of record



Task to Delete

608

Do you want to delete ::608

Order has been deleted successfully

Updated data

	order_id	customer_id	restaurant_id	total_cost	order_time	order_status
0	600	1007	105	80	2022-04-04T09:27:45	Delivered
1	601	1002	101	20	2022-04-04T10:15:22	Delivered
2	602	1004	104	220	2022-04-04T12:15:22	Pending
3	603	1001	105	100	2022-04-04T11:15:22	Pending
4	604	1005	107	240	2022-04-04T14:15:22	Pending
5	605	1006	107	60	2022-04-04T13:15:20	Delivered
6	606	1006	106	220	2022-04-05T12:17:12	Cancelled
7	607	1007	106	40	2022-04-05T14:17:12	Delivered