Indian Institute of Technology Gandhinagar

Databases CS 432

Report: Assignment 2

Authors

22110050	Birudugadda Srivibhav	20110101	Pramod Limbore
22110260	Srivathsa Vamsi Chaturvedula	20110106	Shriyash Mandavekar
22110124	Sriman reddy	20110071	Haikoo Khandor
22110162	Nikhilesh Myanapuri	20110104	Madhav Kanda

Under the Supervision of

Prof. Mayank Singh

February 16, 2024

Responsibility of G1

Github Repository: https://github.com/Sparky1743/Food-Delivery-System-Website

3.1.1: ACID and Constraints:

1. **Atomicity:** In our food delivery system, we ensure that when a delivery person accepts an order, the whole process is like a single package. If something goes wrong (like a payment failure), we can easily undo the entire process and go back to where we started. We make use of the database system's features to handle this, ensuring transactions can be rolled back if there's an error.

2. **Consistency:** Before processing any order, our system checks to make sure everything is in order. For example, we confirm that each order is assigned to only one restaurant by using foreign keys. We also validate that each order has a proper delivery address and accurate payment info, and this is tracked through data definition.

3. **Isolation:** To make sure different tasks happening at the same time don't mess with each other, we use a locking system in our database. When a delivery person takes on an order, the relevant parts of the database get locked until the entire process is done. This is made possible using foreign keys in the respective table.

4. **Durability:** We've made our food delivery system robust by setting up a system that regularly backs up the database to a safe place. Though we haven't put it into action yet, once the real database is in place, we can easily recover it after a system failure. This ensures our food delivery system stands up to the test of time.

Our food delivery management system ensures the dependability and coherence of its database transactions by using the ACID properties. This method makes sure that the data in the database is precise, even when faced with errors or failures.

We have applied these constraints in our database management system to make sure that the data in the database tables follows certain rules and limits. These constraints are present in the SQL dump that we have submitted.

1. ***delivery_rating:*** This constraint ensures that the customer's rating for the delivery is from 0 to 5 (inclusive).
2. ***item_rating:*** This constraint ensures that the customer's rating for the restaurant is from 0 to 5 (inclusive).
3. ***delivery_charges:*** This constraint ensures that an order's delivery charges are not negative (i.e., at least 0).
4. ***amount:*** This constraint makes sure that the amount paid for an order is more than 0.
5. ***tip:*** This constraint ensures that the tip for a delivery is not negative (i.e., at least 0).
6. ***delivery_time:*** This constraint ensures that an order's delivery time is later than the pickup time.
7. ***payment_method:*** This constraint ensures that the payment method used for an order is one of the permitted options (UPI, net banking, Debit Card, or Credit Card).
8. ***order_status:*** This constraint ensures that the order status is one of the permitted options (pending, processing or delivered).
9. ***email, phone and license_id*** are UNIQUE.

By applying these constraints, we ensure that the data in our database is correct, consistent, and valid, which helps enhance our system's overall quality and reliability.

3.1.2: Indexing, user-defined data type, table extensions:

Indexing

- Indexing in a database is employed for search operations to enhance query performance by swiftly locating specific records. By organising data in a structured format, indexing reduces the time and resources required for searching, thereby optimising database retrieval efficiency. This leads to faster data access and improved overall system responsiveness, particularly in large datasets.

- We utilised the "fd beta" database, specifically focusing on the "Food_Item" entity containing approximately 10,000 instances or rows. Our objective was to observe the time difference between conducting searches with and without indexing, particularly targeting the "food_item" attribute by searching for a specific URL. The process is described below through screenshots.

- We used the following commands
    - EXPLAIN ANALYZE
      SELECT * FROM Food_Item WHERE photo_url LIKE "%http://weber.info%";
    - CREATE INDEX idx_food_item_photo_url ON Food_Item(photo_url);
    - EXPLAIN ANALYZE
      SELECT * FROM Food_Item WHERE photo_url LIKE "%http://weber.info%";

- Output of Explain analyze command

```
Database changed
mysql> EXPLAIN ANALYZE SELECT * FROM Food_Item WHERE photo_url LIKE "%http://weber.info%";
+-----------------------------------------------------------------------------------------------------------
| EXPLAIN
+-----------------------------------------------------------------------------------------------------------
| -> Filter: (Food_Item.photo_url like '%http://weber.info%')  (cost=999 rows=1083) (actual time=9.01..9.02 rows=1 loops=1)
    -> Table scan on Food_Item  (cost=999 rows=9751) (actual time=0.191..6.42 rows=10000 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------
1 row in set (0.01 sec)

mysql> CREATE INDEX idx_food_item_photo_url ON Food_Item(photo_url);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT * FROM Food_Item WHERE photo_url LIKE "%http://weber.info%";
+-----------------------------------------------------------------------------------------------------------
| EXPLAIN
+-----------------------------------------------------------------------------------------------------------
| -> Filter: (Food_Item.photo_url like '%http://weber.info%')  (cost=999 rows=1083) (actual time=8.6..8.61 rows=1 loops=1)
    -> Table scan on Food_Item  (cost=999 rows=9751) (actual time=0.131..6.07 rows=10000 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------
1 row in set (0.01 sec)
```

Performing Search	Without Indexing	With Indexing
Execution Duration	0.191 sec	0.131 sec

User-defined data types:

A user-defined data type is one that the user creates instead of the system. It lets users make their own data type. The following code shows how to do this:

```
-- CREATE TYPE details AS (
--  email varchar(50) NOT NULL,
--  phone varchar(50) NOT NULL
-- );
-- As CREATE TYPE is not in mySQL we use JSON datatype

DROP TABLE IF EXISTS Customers;
CREATE TABLE Customers (
  customer_id int(8) NOT NULL,
  first_name varchar(50) NOT NULL,
  middle_name varchar(50) DEFAULT NULL,
  last_name varchar(50) NOT NULL,
  dob date DEFAULT NULL,
  age int(2) DEFAULT NULL,
  contact_details JSON,  -- JSON data type for contact details
  password varchar(50) NOT NULL,
  PRIMARY KEY (customer_id)
) ;

INSERT INTO Customers (customer_id, first_name, middle_name, last_name, dob, age, contact_details, password)
VALUES(1,'Rajesh','Kumar','Sharma','1980-05-10',41,'{"email":"rajesh.sharma@example.com","phone":"9876543210"}','password1');
```

Here, we have set the datatype of “contact_details” to JSON as CREATE TYPE is not in *MySQL*. So, we can now insert any data that is of type JSON. The content of JSON objects can be of any type like numeric, char, varchar, int, etc

Table Extension

In MySQL, extending a table involves augmenting it with additional columns without altering the existing columns or data. Essentially, it's about expanding upon an existing table by incorporating new columns while retaining the original structure intact. This approach proves beneficial when you need to integrate fresh information into an existing table without creating an entirely new one.

There are some methods to achieve the result; they are described below

1. To achieve this, you can utilise the ALTER TABLE table_name ADD COLUMN column_name datatype command to seamlessly add more columns to the extended table. We have added an additional column tip comment for the Delivery table, which is shown below.

   Table before using the command

```
mysql> SELECT * FROM Delivery;
```

order_id	agent_id	delivery_review	delivery_rating	delivery_charges	pickup_time	delivery_time	delivery_status	tip
3557	2457	Porro voluptatum possimus quis. Debitis cupiditate	0.00	183.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	993.00
6403	11855		5.00	26.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	91.00
17874	20525		0.00	191.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	956.00
66537	21888		0.00	136.00	2004-11-29 00:00:00	0000-00-00 00:00:00	deliverd	332.00
74034	22577		5.00	113.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	768.00
89909	29866		1.00	125.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	30.00
112462	32046		0.00	74.00	2004-11-00 00:00:00	0000-00-00 00:00:00	deliverd	661.00
116081	36778	Atque rerum ut omnis consectetur aspernatur ea. Ex	0.00	183.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	978.00
124606	43622		0.00	70.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	507.00

   Command used

```
mysql> ALTER TABLE Delivery ADD COLUMN  tip_comment varchar(255);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   Table after using the command

```
mysql> SELECT * FROM Delivery;
```

order_id	agent_id	delivery_review	delivery_rating	delivery_charges	pickup_time	delivery_time	delivery_status	tip	tip_comment
3557	2457	Porro voluptatum possimus quis. Debitis cupiditate	0.00	183.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	993.00	NULL
6403	11855		5.00	26.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	91.00	NULL
17874	20525		0.00	191.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	956.00	NULL
66537	21888		0.00	136.00	2004-11-29 00:00:00	0000-00-00 00:00:00	deliverd	332.00	NULL
74034	22577		5.00	113.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	768.00	NULL
89909	29866		1.00	125.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	30.00	NULL
112462	32046		0.00	74.00	2004-11-00 00:00:00	0000-00-00 00:00:00	deliverd	661.00	NULL
116081	36778	Atque rerum ut omnis consectetur aspernatur ea. Ex	0.00	183.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	978.00	NULL
124606	43622		0.00	70.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	507.00	NULL
124661	54450	Non amet ex sit sed. Quo facere fuga nam consequun	0.00	78.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	451.00	NULL
126247	56987	Quaerat voluptatem quia ut nihil vero. Rerum inven	0.00	7.00	0000-00-00 00:00:00	0000-00-00 00:00:00	deliverd	817.00	NULL

2. We can use CREATE TABLE Top_Orders as (SELECT * FROM Ordered_items WHERE (Ordered_items.item_rating=5)); This will create a table Top_Orders, which will have all columns from Orders and contain all the orders whose item rating is 5.

   Table before using the command

```
mysql> SELECT * FROM Ordered_items LIMIT 10;
```

order_id	item_id	item_quantity	item_rating	item_review	notes
3557	59	18	3.00	Vero molestiae ex nulla dicta et maxime. Libero re	Esse dolores dolores voluptate et est in et. Illum ratione iste nihil maiores in tempora non.
6403	79	10	1.00	Veniam dolores et corporis iste nisi facere. Offic	
17874	81	16	0.00	Atque at vel rerum accusantium magnam. Earum quos	Laborum aut et et aut et.
66537	240	19	0.00	Tempora dolorum est repellat voluptatum. Libero ac	Adipisci quis vitae magnam quas.
74034	288	20	2.00	Odit omnis delectus qui alias. Dolores aut perfere	Rem voluptates tempore tempora molestiae porro molestiae ipsa voluptatem. Tempore in soluta fugit er
89909	333	16	0.00		
112462	380	4	0.00	Eos perferendis occaecati et nulla velit. Earum es	
116081	383	13	0.00	Porro aut maxime asperiores ratione cumque dolore.	
124606	501	11	1.00	Perspiciatis repudiandae dolorem sed dolor. A eius	Aut deserunt enim accusantium doloremque suscipit dignissimos praesentium maiores. Aut ut dolor in d
124661	543	6	0.00		Numquam qui eum tempore enim nobis quo animi aperiam.

   Command used

```
mysql> CREATE TABLE Top_Orders as (SELECT * FROM Ordered_items  WHERE (Ordered_items.item_rating=5));
Query OK, 859 rows affected (0.05 sec)
Records: 859  Duplicates: 0  Warnings: 0
```

Table after using the command

```
mysql> SELECT * FROM Top_Orders LIMIT 10;
+----------+---------+---------------+-------------+-------------------------------------------+-----------------------------------------------------------------------------------------------------------+
| order_id | item_id | item_quantity | item_rating | item_review                               | notes                                                                                                     |
+----------+---------+---------------+-------------+-------------------------------------------+-----------------------------------------------------------------------------------------------------------+
|   409295 |    3448 |            18 |        5.00 | Quis pariatur est et est qui deserunt. Modi quae r | Recusandae facere eum non non aliquid. Culpa expedita id ad.                                              |
|   763467 |    6286 |            18 |        5.00 |                                           |                                                                                                           |
|   782950 |    6353 |            10 |        5.00 | Est sunt deserunt et earum. Excepturi excepturi al |                                                                                                           |
|   803507 |    6789 |            13 |        5.00 | A commodi fugiat et rerum enim recusandae. Iure se |                                                                                                           |
|   873678 |    7591 |             2 |        5.00 |                                           | Ut cumque quam cum praesentium a. Et voluptates natus laboriosam voluptatem iusto.                        |
|   902805 |    7991 |            19 |        5.00 |                                           | Eos soluta amet velit deleniti eum.                                                                       |
|  1036421 |    9039 |            14 |        5.00 | Et autem explicabo ea odit iusto error. Autem sed |                                                                                                           |
|  1386554 |   12178 |             2 |        5.00 | Suscipit autem voluptatem odit. Magni in dicta ull |                                                                                                           |
|  1399563 |   12529 |             2 |        5.00 | Iusto amet ex sed suscipit expedita voluptatum. Po | Laborum sed similique consectetur nisi explicabo tenetur quis. Minus perspiciatis sit dignissimos po      |
|  1404427 |   12630 |            16 |        5.00 | Distinctio deleniti voluptatem molestias id except |                                                                                                           |
+----------+---------+---------------+-------------+-------------------------------------------+-----------------------------------------------------------------------------------------------------------+
10 rows in set (0.00 sec)
```

3. We can create a new table with an additional column we want to add and then use a foreign key constraint to link the new table to the original.
   The detailed code and results are provided in the image below

```
mysql> CREATE TABLE Restaurant_extended(restaurant_id INT PRIMARY KEY, famous_item varchar(255), FOREIGN KEY (restaurant_id) REFERENCES Restaurant(restaurant_id));
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM Restaurant_extended LIMIT 10;
Empty set (0.00 sec)

mysql> INSERT INTO Restaurant_extended(restaurant_id) SELECT restaurant_id FROM Restaurant;
Query OK, 1000 rows affected (0.02 sec)
Records: 1000  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Restaurant_extended LIMIT 10;
+---------------+-------------+
| restaurant_id | famous_item |
+---------------+-------------+
|           149 | NULL        |
|           171 | NULL        |
|           346 | NULL        |
|           435 | NULL        |
|           564 | NULL        |
|           650 | NULL        |
|           675 | NULL        |
|           920 | NULL        |
|           929 | NULL        |
|           958 | NULL        |
+---------------+-------------+
10 rows in set (0.00 sec)
```

Additional Note:

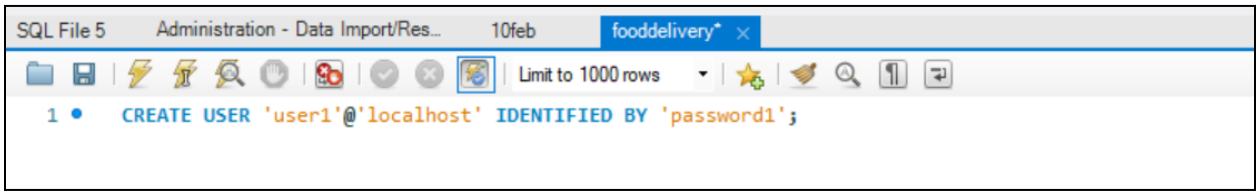
- In type 1, where we used the ALTER command, we will get NULL values to the added columns initially as no data has been inserted into the same and in type 3, the CREATE TABLE statement creates a new table with the same structure as the original table without any data. In other words, the new table is empty, and all the columns have NULL values until you insert data into them. For type 3, the created table Restaurant_extended has the restaurant_id empty initially, but I have inserted the data into that column(Please refer to the figure in type 3 for the same).

- In type 2, you use the CREATE TABLE ... AS SELECT statement to create a new table based on a SELECT query. The SELECT query retrieves data from the Ordered_item table and inserts it into the new table so it contains actual data.

Responsibility of G2

We imported data into the MySQL server, then:

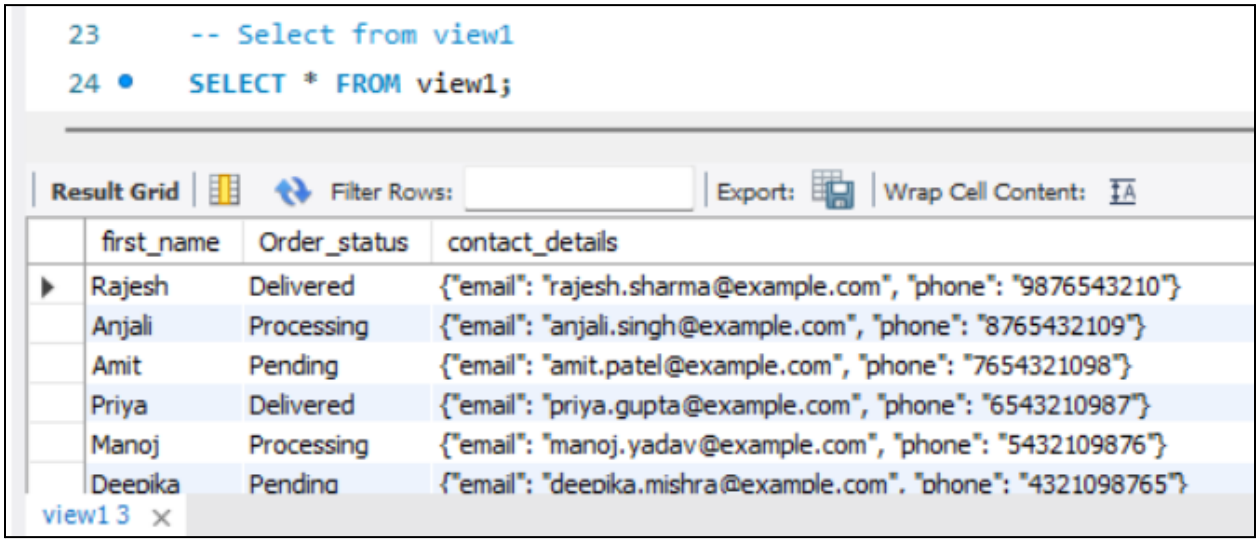
3.2.1 Created a user named "user1" with the password "password1"



3.2.2 We then Created Views on the two tables formed by G1 as view1 and view2.

```
11      -- Create view1
12 •    CREATE VIEW view1 AS
13      SELECT customers.first_name, Orders.Order_status, customers.contact_details
14      FROM Orders
15      JOIN customers ON Orders.customer_id = customers.customer_id;
16
17      -- Create view2
18 •    CREATE VIEW view2 AS
19      SELECT Restaurant.restaurant_name, Food_Item.item_name, Restaurant.contact_details
20      FROM Food_Item
21      JOIN Restaurant ON Food_Item.restaurant_id = Restaurant.restaurant_id;
```

view1 contains columns as first_name, contact_details(column which has user-defined data type) from customers table and Order_status from order table



view2 contains columns as restaurant_name, contact_details(column which has user-defined data type) from Restaurant table and item_namefrom Item table

```
26      -- Select from view2
27 •    SELECT * FROM view2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

restaurant_name	item_name	contact_details
Taj Mahal Restaurant	Paneer Tikka	{"email": "tajmahal@example.com", "phone": "+91 9876543210"}
Spice Garden	Chicken Biryani	{"email": "spicegarden@example.com", "phone": "+91 8765432109"}
Punjabi Dhaba	Butter Chicken	{"email": "punjabidhaba@example.com", "phone": "+91 7654321098"}
Southern Spice	Masala Dosa	{"email": "southernspice@example.com", "phone": "+91 6543210987"}
The Mughal Feast	Mutton Rogan Josh	{"email": "mughalfeast@example.com", "phone": "+91 5432109876"}
Coastal Curry House	Fish Curry	{"email": "coastalcurry@example.com", "phone": "+91 4321098765"}

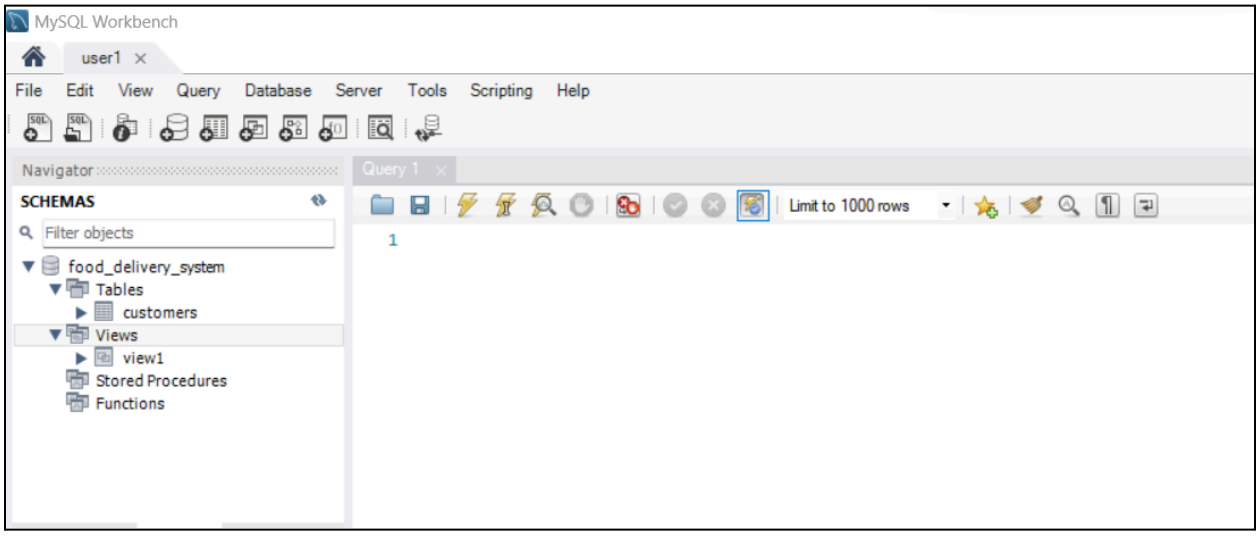
view2 4

3.2.3 & 3.2.4 Granted "user1" the following permissions on the customers table: SELECT, UPDATE, DELETE

Granted "user1" the following permissions on view1: SELECT

```
29      -- Grant permissions to 'user1'
30 •    GRANT SELECT, UPDATE, DELETE ON customers TO 'user1'@'localhost';
31 •    GRANT SELECT ON view1 TO 'user1'@'localhost';
```

For user1, you can see only the customers' table and view1 as follows:



3.2.5 Operations on customers table as user1:

- SELECT:

```
use food_delivery_system;
SELECT * FROM customers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	first_name	middle_name	last_name	dob	age	contact_details	password
1	Rajesh	Kumar	Sharma	1980-05-10	41	{"email": "rajesh.sharma@example.com", "phon...	password1
2	Anjali	NULL	Singh	1995-08-20	26	{"email": "anjali.singh@example.com", "phone":...	password2
3	Amit	Kumar	Patel	1988-03-15	33	{"email": "amit.patel@example.com", "phone": "...	password3
4	Priya	NULL	Gupta	1992-11-25	29	{"email": "priya.gupta@example.com", "phone":...	password4
5	Manoj	Kumar	Yadav	1982-07-08	39	{"email": "manoj.yadav@example.com", "phone...	password5
6	Deepika	NULL	Mishra	1987-09-18	34	{"email": "deepika.mishra@example.com", "pho...	password6

- UPDATE:

```
use food_delivery_system;
SELECT * FROM customers;

UPDATE customers SET password = 'raju1234' WHERE customer_id=1;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:19:50	UPDATE customers SET password = 'raju1234' WHERE customer_id=1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec

- DELETE:

Here, when attempting to delete the customer with customer_id = 15, an error occurs due to the usage of customer_id as a foreign key in the 'orders' table without specifying 'ON DELETE CASCADE' for that foreign key constraint.

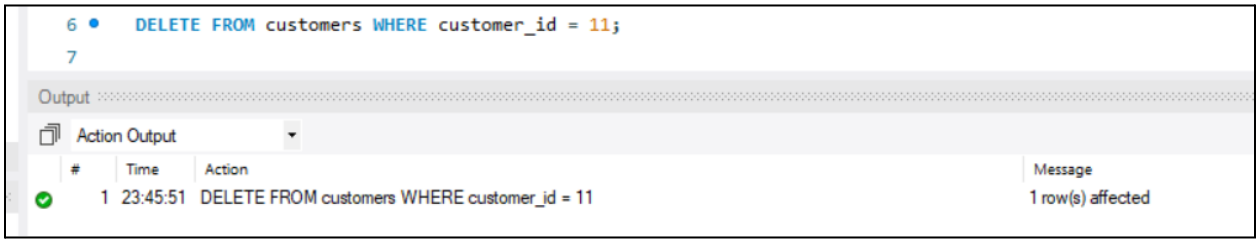
```
DELETE FROM customers WHERE customer_id = 15;
```

Output

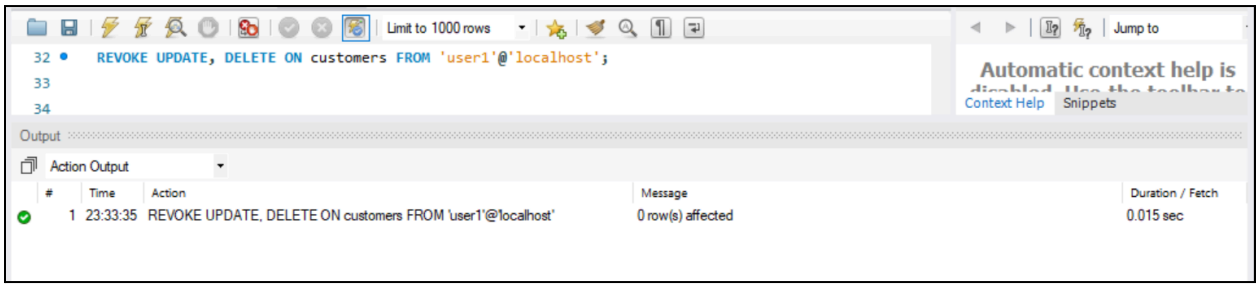
Action Output

#	Time	Action	Message
1	23:03:02	DELETE FROM customers WHERE customer_id = 15	Error Code: 1217. Cannot delete or update a parent row: a foreign key constraint fails

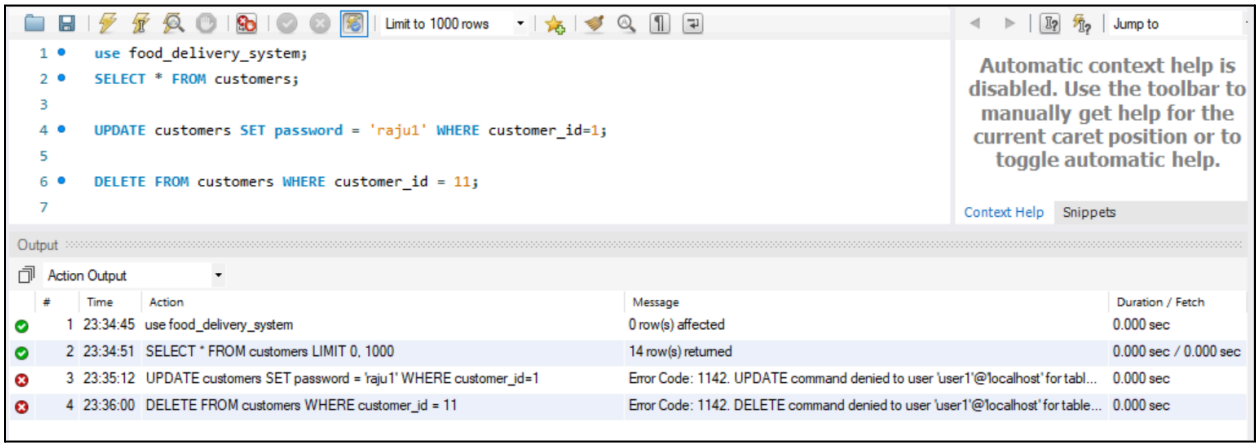
After specifying the 'ON DELETE CASCADE' as that foreign key constraint, we are able to perform the delete operation



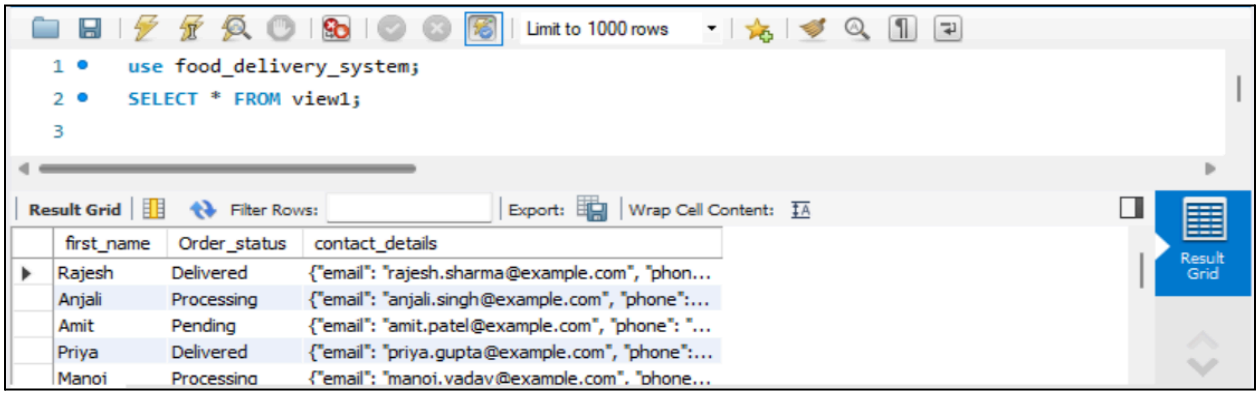
3.2.6 For revoking the permission for user1 on the customers table, we can use



After revoking the permission to update and delete, the user can not update and delete the customers table.

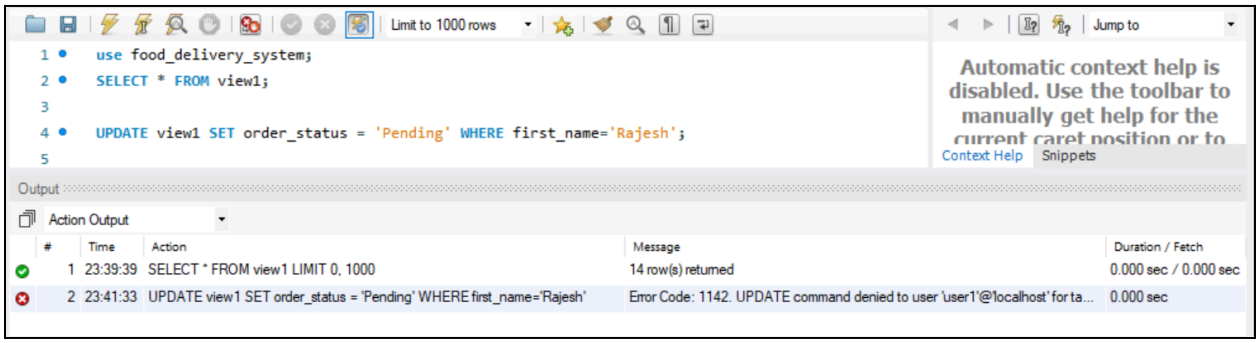


3.2.7 SELECT command on view1



UPDATE command on view1

The user cannot update entries in view1 as they do not have permission to update the underlying tables in the database.

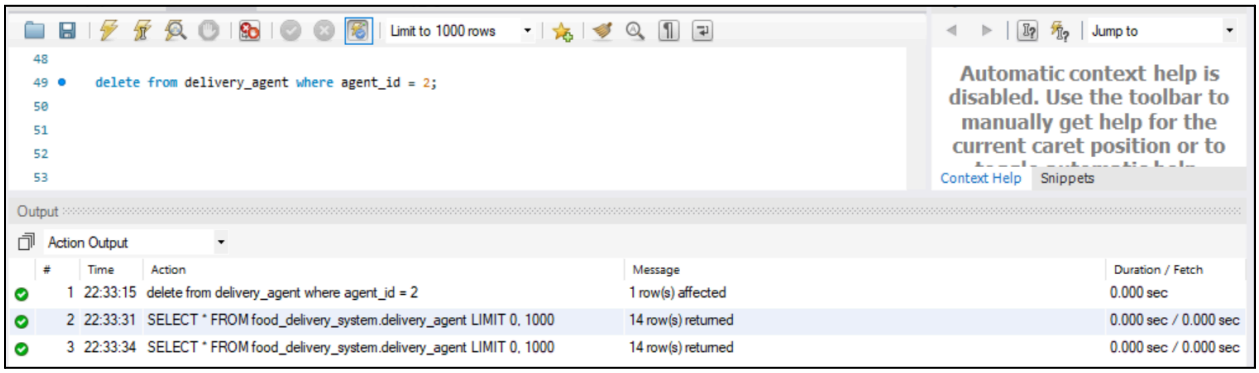


DELETE command on view1

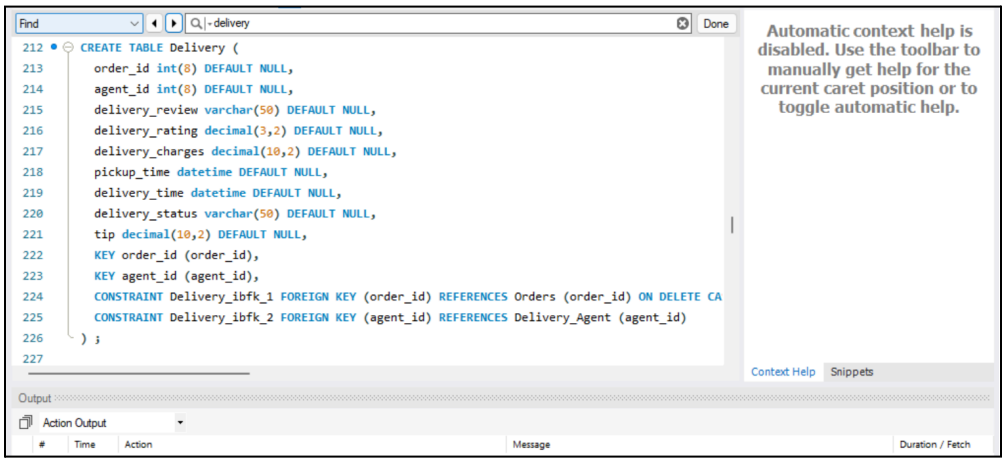
The user cannot delete entries from view1 as they do not have permission to delete from the underlying tables in the database.

Referential integrity:

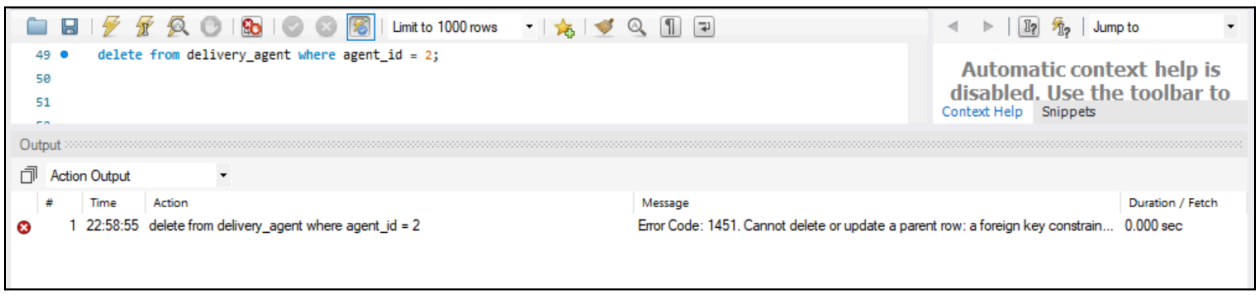
1. If we try to delete the delivery agent from the database, it would leave that specific delivery without assigning it to another delivery agent.



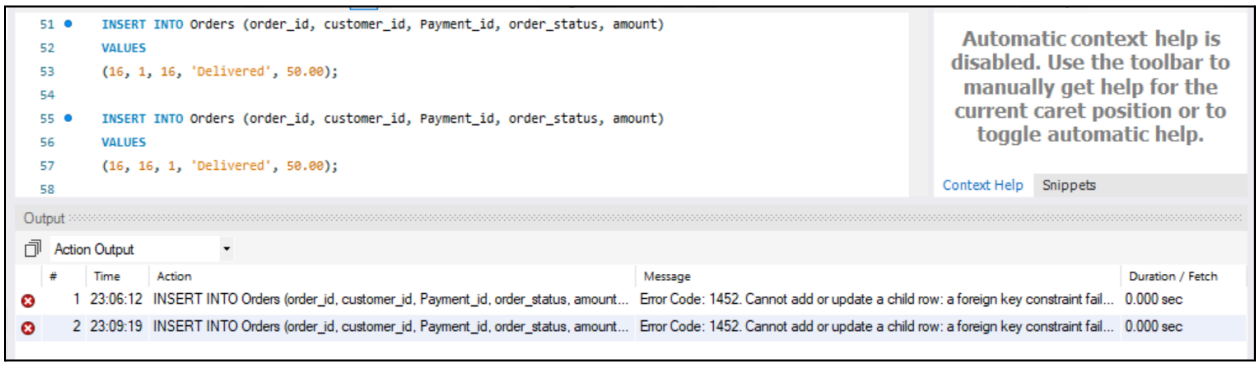
Solution: We can tackle this problem by not including '**ON DELETE CASCADE**' in the delivery table where delivery_agent is a foreign key.



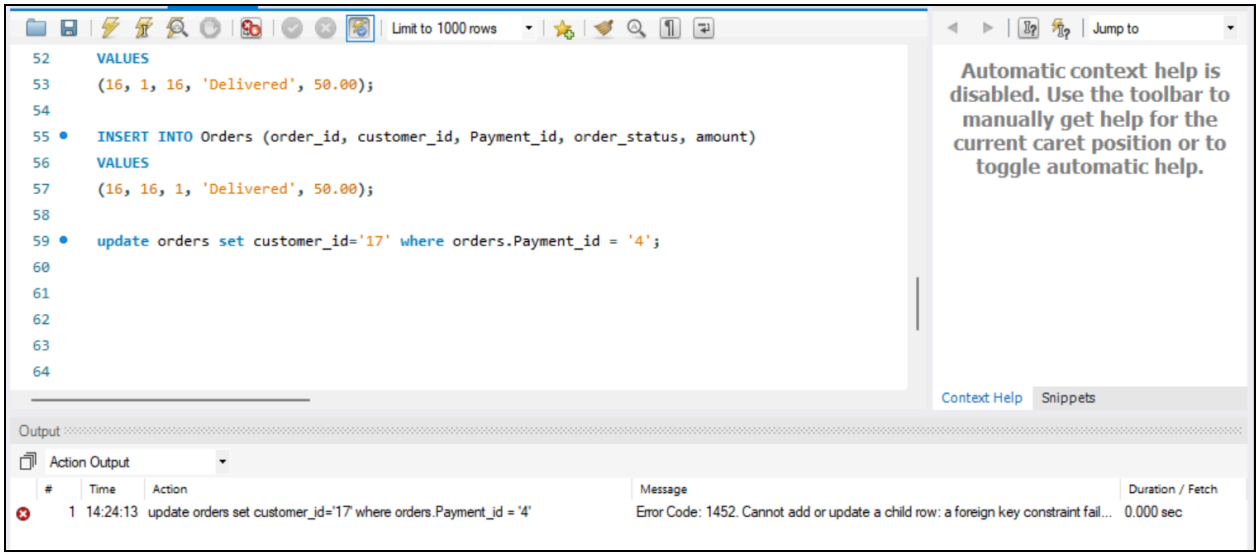
After that, if we try to delete the delivery_agent, it will result in an error saying that the foreign constraint fails.



2. If you try to add an order without a valid payment_id or customer_id, this will violate the referential integrity between the order and customer tables.



3. If you try to update the order with an invalid customer_id, it will violate the referential integrity between the customer and the orders table.



4. Delivery_rating should be between 0 and 5. If any user tries to update the rating to greater than five, the database will deny the request.
Solution: To Include the above constraint for delivery_rating, we need to include.
CONSTRAINT CHK_delivery_rating_range CHECK (delivery_rating >= 0 AND delivery_rating <= 5)

```
update orders set customer_id='17' where orders.Payment_id = '4';

update delivery set delivery_rating = 6 where order_id = 3;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Context Help | Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	14:41:38	update delivery set delivery_rating = 6 where order_id = 3	Error Code: 3819. Check constraint 'CHK_delivery_rating_range' is violated.	0.000 sec

5. Another constraint is that the delivery_time should always be after pickup.

```
update orders set customer_id='17' where orders.Payment_id = '4';

update delivery set delivery_rating = 6 where order_id = 3;

INSERT INTO Delivery (order_id, agent_id, delivery_review, delivery_rating, delivery_charges,
VALUES
(16, 1, 'Good service', 4, 5.00, '2024-02-14 12:30:00', '2024-02-14 12:00:00', 'Delivered', 2
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Context Help | Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	14:41:38	update delivery set delivery_rating = 6 where order_id = 3	Error Code: 3819. Check constraint 'CHK_delivery_rating_range' is violated.	0.000 sec
2	14:45:56	INSERT INTO Delivery (order_id, agent_id, delivery_review, delivery_rating, deli...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fail...	0.000 sec

6. Similarly, restaurant_rating should be less than 5 and greater than 0.

7. Order_status can only take values from ‘Delivered’, ‘Processing’, and ‘Pending’.
Solution: **order_status ENUM('Delivered', 'Processing', 'Pending') NOT NULL**

```
INSERT INTO Delivery (order_id, agent_id, delivery_review, delivery_rating, delivery_charges,
VALUES
(16, 1, 'Good service', 4, 5.00, '2024-02-14 12:30:00', '2024-02-14 12:00:00', 'Delivered', 2

update orders set order_status = 'Remaining' where order_id = 2;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

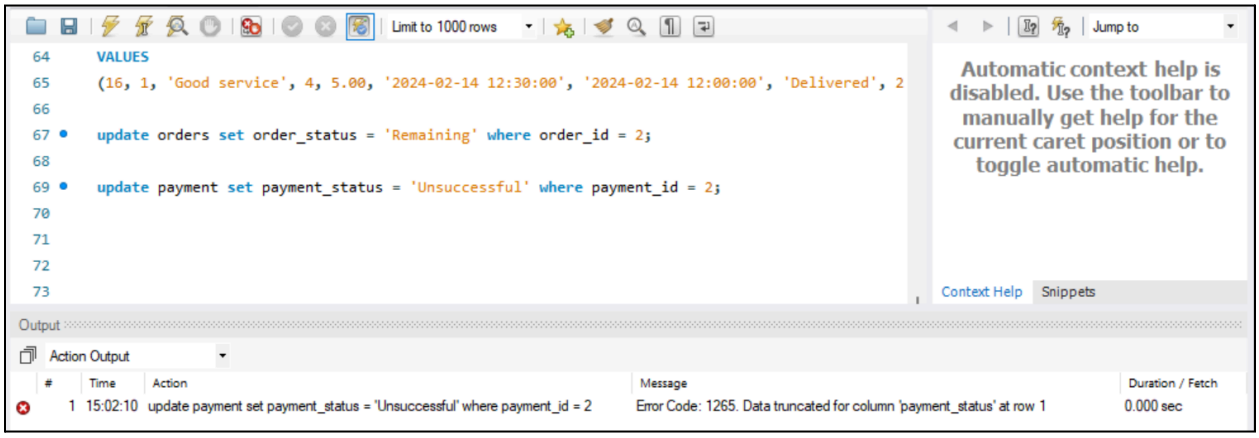
Context Help | Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	14:57:30	update orders set order_status = 'Remaining' where order_id = 2	Error Code: 1265. Data truncated for column 'order_status' at row 1	0.000 sec

8. Payment_status can only take values from ‘Successful’, ‘Failed’ or ‘Pending’. Updating a payment with an invalid payment status will result in an error.
Solution: **payment_status ENUM('Successful', 'Failed', 'Pending') NOT NULL**



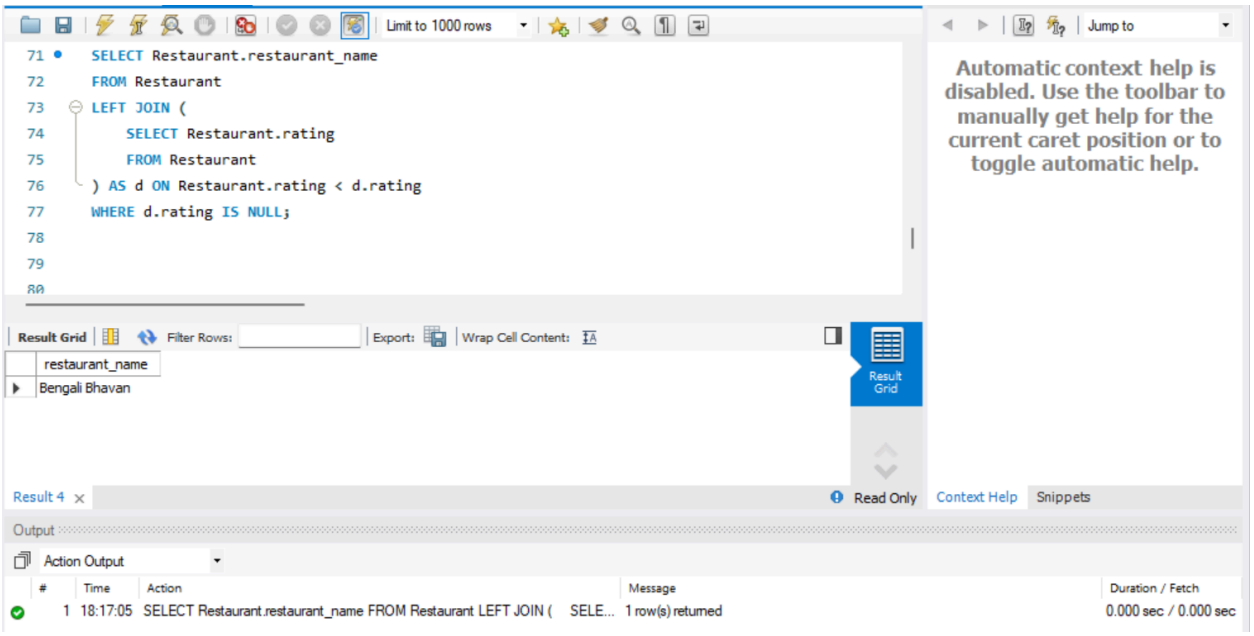
Responsibility of G1 and G2

3.3.1 Listing out the restaurant with the highest rating

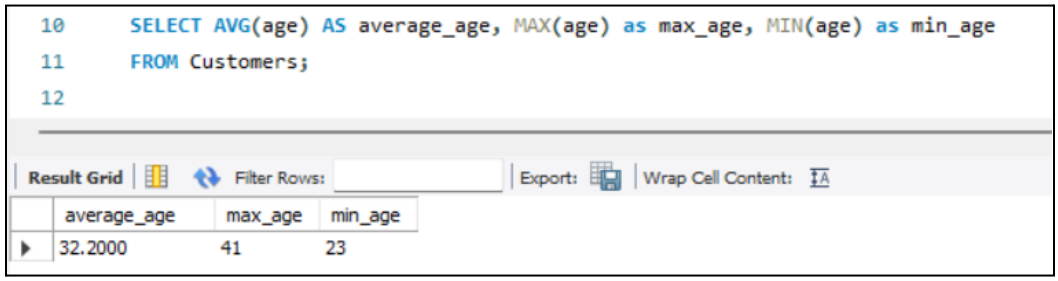
Relational algebra:

$$\pi_{restaurant\_name} - \pi_{Restaurant.restaurant\_name}(\sigma_{Restaurant.rating < d.rating}(Restaurant \times \rho_{d} Restaurant))$$

To calculate the restaurant with the highest rating, we first have to compute a temporary relation consisting of restaurants that do not have the largest rating. Then, we take the set difference between the relation restaurant name and the temporary relation.



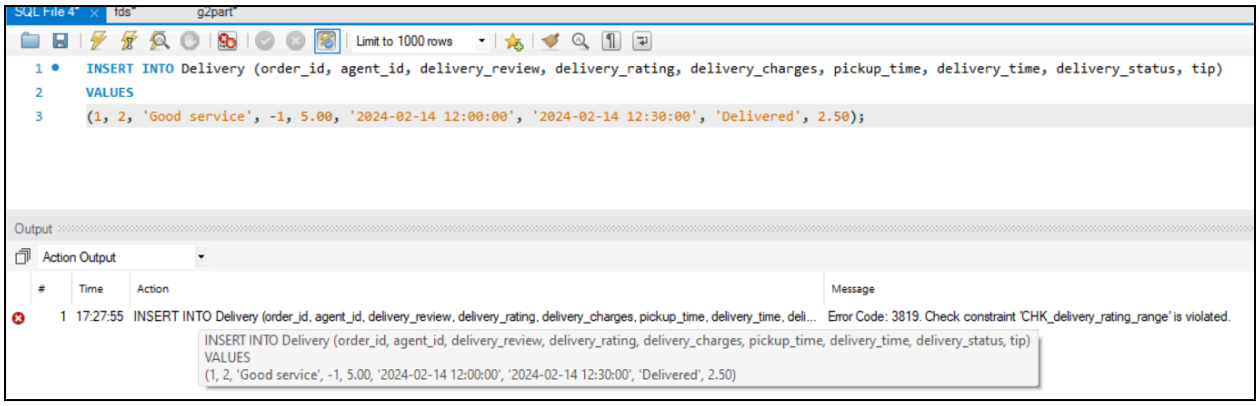
3.3.2 Average, max and min age of customers:



Relational algebra:

$$\pi_{average\_age,max\_age,min\_age}(\gamma_{AVG(age),MAX(age),MIN(age)}(Customers))$$

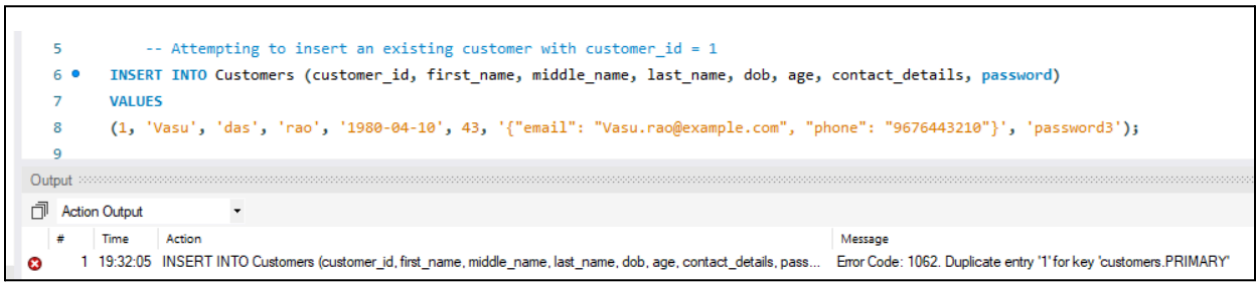
3.3.3 Insert an entry for Delivery with negative delivery_rating on which our database throws an error:



Relational algebra:

Delivery←Delivery ∪ {(1,2,'Good service',−1,5.00,'2024-02-14 12:00:00','2024-02-14 12:30:00','Delivered',2.50)}

3.3.4 Attempting to insert an existing customer with customer_id = 1



Relational algebra:

$$Customers \leftarrow Customers \cup \{1, 'Vasu', 'das', 'rao', '1980-04-10', 43, \\ '\{"email": "Vasu.rao@example.com", "phone": "9676443210"\}', 'password3\}$$

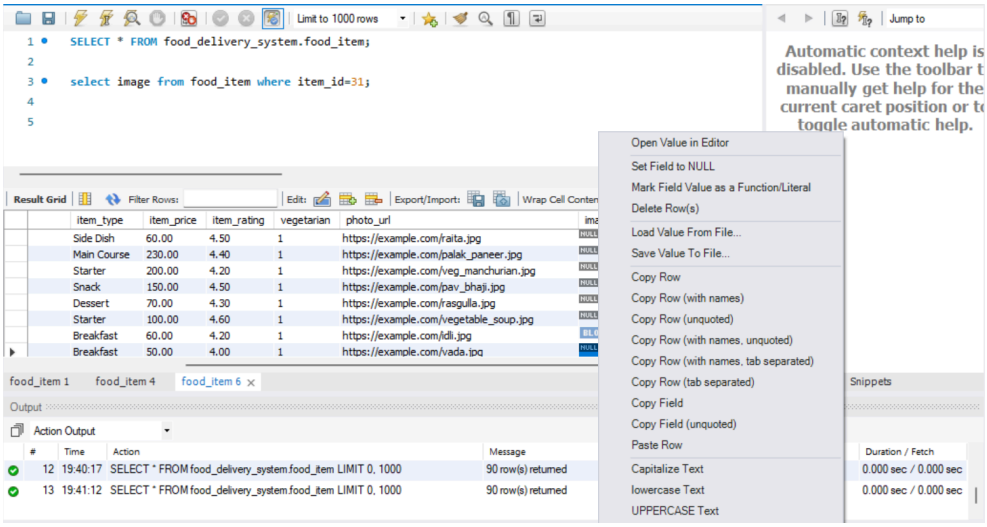
This expression represents the union of the existing Customers relation with a new tuple containing the specified attribute values, effectively inserting a new record into the Customers table.

3.3.5 Insert an image into the MySQL database.

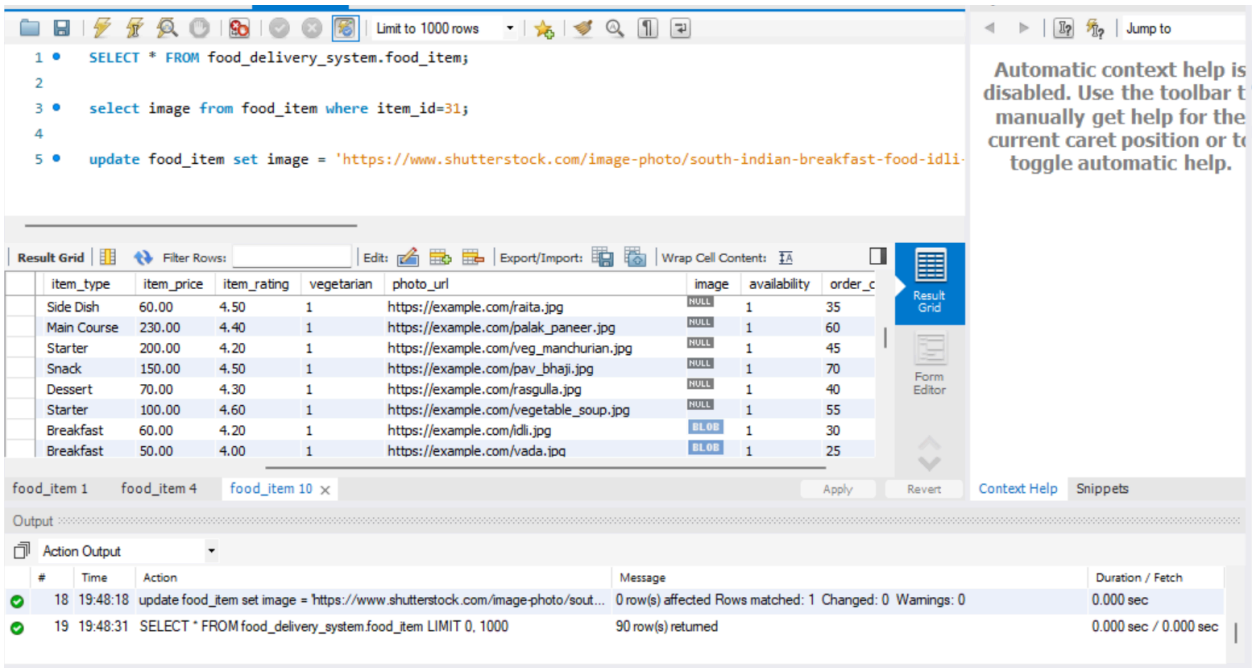
The image's data type must be BLOB for storing images in MySQL.BLOB stands for “Binary Large Object” and represents a database type to store binary data. It stores images, videos, audio, or any other data type that cannot be represented using text. Depending on the size of the image, the data type can be blob or longblob.

```
CREATE TABLE Food_Item (
  item_id int(8) NOT NULL,
  restaurant_id int(8) NOT NULL,
  item_name varchar(50) NOT NULL,
  item_type varchar(50) NOT NULL,
  item_price decimal(10,2) NOT NULL,
  item_rating decimal(3,2) DEFAULT NULL,
  vegetarian tinyint(1) NOT NULL,
  photo_url varchar(100) DEFAULT NULL,
  image longblob,
  availability tinyint(1) NOT NULL,
  order_count int(8) NOT NULL,
  PRIMARY KEY (item_id),
  KEY restaurant_id (restaurant_id),
  CONSTRAINT Food_Item_ibfk_1 FOREIGN KEY (restaurant_id) REFERENCES Restaurant (restaurant_id) ON DE
);
```

We have stored the image referring to the article provided in the assignment. You can also add captions by editing the fields.



Another way to add the image is by using the update command with the image parameter as a link to the image.



3.3.6 Finding statistical information about food items:

It includes average, minimum, and maximum prices, along with the corresponding restaurants for each extreme price.

***Relational algebra*:**

$$Item\_statistics\_with\_joints \leftarrow \rho_{item\_name,\ avg\_price,\ min\_price,\ max\_price,\ min\_price\_item\_id,min\_price\_restaurant,\ max\_price\_item\_id,\ max\_price\_restaurant}(\sigma_{(Item\_statistics.item\_name = Food\_Item.item\_name\ AND\ Item\_statistics.min\_price = Food\_Item.item\_price)}(Item\_statistics \bowtie Food\_Item \bowtie Restaurant \bowtie \rho_{(min\_price\_item\_id,\ restaurant\_id,\ item\_name,\ min\_price\_restaurant}(Food\_Item) \bowtie \rho_{max\_price\_item\_id,\ restaurant\_id,\ item\_name,\ max\_price\_restaurant}(Food\_Item))$$

```sql
CREATE TABLE Item_statistics AS
SELECT distinct
    item_name,
    AVG(item_price) OVER(PARTITION BY item_name) AS avg_price,
    MIN(item_price) OVER(PARTITION BY item_name) AS min_price,
    MAX(item_price) OVER(PARTITION BY item_name) AS max_price
FROM
    Food_Item;

SELECT distinct
    i.item_name,
    i.avg_price,
    i.min_price,
    i.max_price,
    r_min.restaurant_name AS min_price_restaurant,
    r_max.restaurant_name AS max_price_restaurant
FROM
    Item_statistics i
JOIN
    Food_Item f_min ON i.item_name = f_min.item_name AND i.min_price = f_min.item_price
JOIN
    Restaurant r_min ON f_min.restaurant_id = r_min.restaurant_id
JOIN
    Food_Item f_max ON i.item_name = f_max.item_name AND i.max_price = f_max.item_price
JOIN
    Restaurant r_max ON f_max.restaurant_id = r_max.restaurant_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

item_name	avg_price	min_price	max_price	min_price_restaurant	max_price_restaurant
Vegetable Biryani	226.000000	210.00	240.00	Southern Spice	Spice Garden
Aloo Paratha	118.000000	100.00	130.00	The Mughal Feast	Spice Garden
Dal Makhani	201.000000	180.00	220.00	Spice Garden	Coastal Curry House
Chicken Tikka	272.000000	250.00	300.00	Spice Garden	Rajasthani Delight
Vegetable Biryani	226.000000	210.00	240.00	Southern Spice	Rajasthani Delight

3.3.7 Based on the average price of the item we classified the items into High and medium categories.

```sql
START TRANSACTION;
CREATE TABLE Item_Statistics AS
SELECT
    item_name,
    MAX(item_price) AS max_price,
    MIN(item_price) AS min_price,
    AVG(item_price) AS avg_price,
    CASE
        WHEN AVG(item_price) > 50 THEN 'High'
        WHEN AVG(item_price) > 25 THEN 'Medium'
        ELSE 'Low'
    END AS price_category
FROM
    Food_Item
GROUP BY
    item_name;
SELECT * FROM Item_Statistics;

-- Assuming everything is successful Commit command is used
COMMIT;
```

Limit to 1000 rows | Jump to

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

Context Help | Snippets

Work Distribution

Team Member	Contribution
1. Birudugadda Srivibhav(G1)	Generation of random data, Putting up the necessary constraints. Making up the final report. Implementing indexing, user-defined data types, and table extensions, Generation of random data
2. Srivathsa Vamsi(G1)	Generation of random data, Putting up the necessary constraints. Making up the final report. Implementing indexing, user-defined data types, and table extensions, Generation of random data
3. Sriman Reddy(G1)	Generation of random data, Putting up the necessary constraints. Making up the final report. Implementing indexing, user-defined data types, and table extensions, Generation of random data
4. Nikhilesh Myanapuri(G1)	Generation of random data, Putting up the necessary constraints. Making up the final report. Implementing indexing, user-defined data types, and table extensions, Generation of random data
5. Haikoo Khandor(G2)	Implementing indexing, user-defined data types, and table extensions. Creating a user, granting different permission on tables and views
6. Madhav Kanda(G2)	Situation where referential integrity is violated, updates in the table and how we can solve such problems
7. Pramod Limbore(G2)	Write up five operations and their corresponding tuple relational calculus/ relational algebra and SQL queries, Adding up the photo into the database.
8. Shriyash Mandavekar(G2)	Write up five operations and their corresponding tuple relational calculus/ relational algebra and SQL queries, Adding up the photo into the database.

Acknowledgements

We want to express our thanks to Prof Mayank Singh and TA Ritesh Patidar for their support in the assignment.

References

- https://forums.mysql.com/read.php?20,17671,27914
- https://www.geeksforgeeks.org/cardinality-in-dbms/?ref=ml_lbp
- https://www.geeksforgeeks.org/design-restaurant-management-system-system-design/#database-design
- https://www.gleek.io/blog/relational-schema
- Classroom Slides