

DBMS ASSIGNMENT

Nikki Gautam

Question 2:

Language Used (for querying): **SQL**

Part A:

For the ER Model created in the earlier question, create the database tables, and normalize them. You are free to modify the original ER diagram with new findings / changes. You are also free to leave some tables in a denormalized state if you can justify it in the final report (part C).

Database Created: fooddelivery

The screenshot shows a database management interface. At the top, there is a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar, two SQL queries are listed:

- 1 • `create database fooddelivery;`
- 2 • `show databases;`

Below the queries, there is a 'Result Grid' section. It includes a 'Filter Rows' search bar and an 'Export' button. The result grid displays a list of databases:

Database
airportdb
busmanagement
fooddelivery
information_schema
mysql
performance_schema
sys

On the right side of the interface, there is a vertical sidebar with icons for 'Result Grid', 'Form Editor', and 'Field Types'.

Database Tables:

1. User table

Used to store information of the users who interact with the system

```
CREATE TABLE User (  
    userid INT PRIMARY KEY,  
    name VARCHAR(100),  
    email VARCHAR(100) UNIQUE,  
    phone VARCHAR(15),  
    address TEXT,  
    usertype ENUM('Customer', 'RestaurantOwner',  
'DeliveryPartner')  
);
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The main area displays a SQL script with line numbers 1 through 12. The script includes a 'use fooddelivery;' statement, a 'CREATE TABLE User' statement with columns: userid (INT PRIMARY KEY), name (VARCHAR(100)), email (VARCHAR(100) UNIQUE), phone (VARCHAR(15)), address (TEXT), and usertype (ENUM('Customer', 'RestaurantOwner', 'DeliveryPartner')), followed by a 'describe User;' statement. Below the script, a 'Result Grid' tab is active, showing a table with 7 columns: Field, Type, Null, Key, Default, Extra, and an empty column. The rows list the table's structure: userid (int, NO, PRI, NULL), name (varchar(100), YES, NULL), email (varchar(100), YES, UNI, NULL), phone (varchar(15), YES, NULL), address (text, YES, NULL), usertype (enum('Customer','RestaurantOwner','DeliveryPa...', YES, NULL), and an empty row at the bottom. On the right sidebar, there are buttons for 'Result Grid', 'Form Editor', and 'Table Editor'.

```
1 • use fooddelivery;  
2  
3 • CREATE TABLE User (  
4     userid INT PRIMARY KEY,  
5     name VARCHAR(100),  
6     email VARCHAR(100) UNIQUE,  
7     phone VARCHAR(15),  
8     address TEXT,  
9     usertype ENUM('Customer', 'RestaurantOwner', 'DeliveryPartner')  
10 );  
11  
12 • describe User;
```

Field	Type	Null	Key	Default	Extra	
userid	int	NO	PRI	NULL		
name	varchar(100)	YES		NULL		
email	varchar(100)	YES	UNI	NULL		
phone	varchar(15)	YES		NULL		
address	text	YES		NULL		
usertype	enum('Customer','RestaurantOwner','DeliveryPa...	YES		NULL		

2. Restaurant table

Used to store information of the Restaurant in consideration

```
CREATE TABLE Restaurant (  
    restaurantid INT PRIMARY KEY,  
    name VARCHAR(100),  
    location VARCHAR(255),  
    cuisine VARCHAR(100),  
    rating DECIMAL(3, 2),  
    contact VARCHAR(15),  
    ownerid INT,  
    FOREIGN KEY (ownerid) REFERENCES User(userid)  
);
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The main editor displays the SQL query for creating the 'Restaurant' table and a 'describe' command. Below the editor, the 'Result Grid' is visible, showing the table's structure with columns: Field, Type, Null, Key, Default, and Extra. The table has seven fields: restaurantid (int, primary key), name (varchar(100)), location (varchar(255)), cuisine (varchar(100)), rating (decimal(3,2)), contact (varchar(15)), and ownerid (int, foreign key to User table).

```
1 • CREATE TABLE Restaurant (  
2     restaurantid INT PRIMARY KEY,  
3     name VARCHAR(100),  
4     location VARCHAR(255),  
5     cuisine VARCHAR(100),  
6     rating DECIMAL(3, 2),  
7     contact VARCHAR(15),  
8     ownerid INT,  
9     FOREIGN KEY (ownerid) REFERENCES User(userid)  
10 );  
11  
12 • describe Restaurant;
```

Field	Type	Null	Key	Default	Extra
restaurantid	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
location	varchar(255)	YES		NULL	
cuisine	varchar(100)	YES		NULL	
rating	decimal(3,2)	YES		NULL	
contact	varchar(15)	YES		NULL	
ownerid	int	YES	MUL	NULL	

3. MenuItem table

Used to store details of dishes available to the customers

```
CREATE TABLE MenuItem (  
    itemid INT PRIMARY KEY,  
    restaurantid INT,  
    name VARCHAR(100),  
    price DECIMAL(10, 2),  
    preparationtime INT,  
    availability BOOLEAN,  
    category VARCHAR(100),  
    FOREIGN KEY (restaurantid) REFERENCES  
Restaurant(restaurantid)  
);
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The main area displays a SQL query with line numbers 1 through 12. The query creates a 'MenuItem' table with columns: itemid (INT PRIMARY KEY), restaurantid (INT), name (VARCHAR(100)), price (DECIMAL(10, 2)), preparationtime (INT), availability (BOOLEAN), and category (VARCHAR(100)). It also includes a foreign key constraint for restaurantid referencing the Restaurant table. Line 12 contains the command 'describe MenuItem;'. Below the query editor, a 'Result Grid' tab is active, showing a table with 7 columns: Field, Type, Null, Key, Default, Extra, and an empty 'Extra' column. The rows list the table's structure: itemid (int, NO, PRI, NULL), restaurantid (int, YES, MUL, NULL), name (varchar(100), YES, NULL), price (decimal(10,2), YES, NULL), preparationtime (int, YES, NULL), availability (tinyint(1), YES, NULL), and category (varchar(100), YES, NULL). On the right sidebar, there are buttons for 'Result Grid', 'Form Editor', and a search icon.

```
1 • CREATE TABLE MenuItem (  
2     itemid INT PRIMARY KEY,  
3     restaurantid INT,  
4     name VARCHAR(100),  
5     price DECIMAL(10, 2),  
6     preparationtime INT,  
7     availability BOOLEAN,  
8     category VARCHAR(100),  
9     FOREIGN KEY (restaurantid) REFERENCES Restaurant(restaurantid)  
10 );  
11  
12 • describe MenuItem;
```

Field	Type	Null	Key	Default	Extra
itemid	int	NO	PRI	<input type="text" value="NULL"/>	
restaurantid	int	YES	MUL	<input type="text" value="NULL"/>	
name	varchar(100)	YES		<input type="text" value="NULL"/>	
price	decimal(10,2)	YES		<input type="text" value="NULL"/>	
preparationtime	int	YES		<input type="text" value="NULL"/>	
availability	tinyint(1)	YES		<input type="text" value="NULL"/>	
category	varchar(100)	YES		<input type="text" value="NULL"/>	

4. Order table

Used to store information corresponding to orders registered

```
CREATE TABLE `Order` (  
   orderid INT PRIMARY KEY,  
    userid INT,  
    restaurantid INT,  
    orderdate DATE,  
    totalamount DECIMAL(10, 2),  
    discountapplied INT,  
    paymentstatus ENUM('Pending', 'Completed',  
'Failed'),  
    deliverypartnerid INT,  
    FOREIGN KEY (userid) REFERENCES User(userid),  
    FOREIGN KEY (restaurantid) REFERENCES  
Restaurant(restaurantid),  
    FOREIGN KEY (discountapplied) REFERENCES  
Discount(discountid),  
    FOREIGN KEY (deliverypartnerid) REFERENCES  
DeliveryPartner(deliverypartnerid)  
);
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The main editor displays the SQL query for creating the 'Order' table, with line numbers 1 through 15. The query is as follows:

```
1 CREATE TABLE `Order` (  
2    orderid INT PRIMARY KEY,  
3     userid INT,  
4     restaurantid INT,  
5     orderdate DATE,  
6     totalamount DECIMAL(10, 2),  
7     discountapplied INT,  
8     paymentstatus ENUM('Pending', 'Completed', 'Failed'),  
9     deliverypartnerid INT,  
10    FOREIGN KEY (userid) REFERENCES User(userid),  
11    FOREIGN KEY (restaurantid) REFERENCES Restaurant(restaurantid),  
12    FOREIGN KEY (discountapplied) REFERENCES Discount(discountid),  
13    FOREIGN KEY (deliverypartnerid) REFERENCES DeliveryPartner(deliverypartnerid)  
14 );  
15 describe `Order`;
```

Below the editor, the 'Result Grid' tab is active, showing the table structure. The grid has columns for Field, Type, Null, Key, Default, and Extra. The data is as follows:

Field	Type	Null	Key	Default	Extra
orderid	int	NO	PRI	NULL	
userid	int	YES	MUL	NULL	
restaurantid	int	YES	MUL	NULL	
orderdate	date	YES		NULL	
totalamount	decimal(10,2)	YES		NULL	
discountapplied	int	YES	MUL	NULL	
paymentstatus	enum('Pending','Completed','Failed')	YES		NULL	
deliverypartnerid	int	YES	MUL	NULL	

On the right side of the interface, there are buttons for 'Result Grid' and 'Form Editor'.

5. Discount table

Used to store information regarding discounts available to the customers

```
CREATE TABLE Discount (  
    discountid INT PRIMARY KEY,  
    description TEXT,  
    percentage DECIMAL(5, 2),  
    code VARCHAR(50)  
);
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a "Limit to 1000 rows" dropdown. The main editor displays the following SQL code:

```
1 • CREATE TABLE Discount (  
2     discountid INT PRIMARY KEY,  
3     description TEXT,  
4     percentage DECIMAL(5, 2),  
5     code VARCHAR(50)  
6 );  
7  
8 • describe Discount;
```

Below the editor, the "Result Grid" tab is active, showing the table structure for the "Discount" table. The grid has columns for Field, Type, Null, Key, Default, and Extra. The data is as follows:

Field	Type	Null	Key	Default	Extra
discountid	int	NO	PRI	HULL	
description	text	YES		HULL	
percentage	decimal(5,2)	YES		HULL	
code	varchar(50)	YES		HULL	

On the right side of the interface, there is a vertical toolbar with icons for "Result Grid", "Form Editor", "Field Types", and "Query".

6. DeliveryPartner table

Used to represent delivery partner information in association with the restaurant

```
CREATE TABLE DeliveryPartner (  
    deliverypartnerid INT PRIMARY KEY,  
    name VARCHAR(100),  
    phone VARCHAR(15),  
    vehicledetails VARCHAR(255)  
);
```

The screenshot shows a database management interface. The top section displays the SQL query for creating the 'DeliveryPartner' table. The bottom section shows the 'Result Grid' with the table's structure.

SQL Query:

```
1 • CREATE TABLE DeliveryPartner (  
2     deliverypartnerid INT PRIMARY KEY,  
3     name VARCHAR(100),  
4     phone VARCHAR(15),  
5     vehicledetails VARCHAR(255)  
6 );  
7  
8 • describe DeliveryPartner;
```

Result Grid:

Field	Type	Null	Key	Default	Extra
deliverypartnerid	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
phone	varchar(15)	YES		NULL	
vehicledetails	varchar(255)	YES		NULL	

The interface includes a toolbar at the top with various icons and a 'Limit to 1000 rows' dropdown. The bottom right sidebar contains icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query'.

7. Feedback table

Used to keep a record of customer feedback corresponding to the orders

```
CREATE TABLE Feedback (  
    feedbackid INT PRIMARY KEY,  
    userid INT,  
    orderid INT,  
    rating DECIMAL(2, 1),  
    comments TEXT,  
    FOREIGN KEY (userid) REFERENCES User(userid),  
    FOREIGN KEY (orderid) REFERENCES `Order`(orderid)  
);
```

The screenshot shows a database management interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
1 • CREATE TABLE Feedback (  
2     feedbackid INT PRIMARY KEY,  
3     userid INT,  
4     orderid INT,  
5     rating DECIMAL(2, 1),  
6     comments TEXT,  
7     FOREIGN KEY (userid) REFERENCES User(userid),  
8     FOREIGN KEY (orderid) REFERENCES `Order`(orderid)  
9 );  
10  
11 • describe Feedback;
```

Below the editor is a 'Result Grid' section with a search bar and an 'Export' button. The grid displays the following data:

Field	Type	Null	Key	Default	Extra
feedbackid	int	NO	PRI	NULL	
userid	int	YES	MUL	NULL	
orderid	int	YES	MUL	NULL	
rating	decimal(2,1)	YES		NULL	
comments	text	YES		NULL	

On the right side, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', and a search icon.

8. Payment table

Used to store information of payments made

```
CREATE TABLE Payment (  
    paymentid INT PRIMARY KEY,  
    orderid INT,  
    paymentmethod ENUM('Card', 'UPI', 'Wallet',  
'NetBanking'),  
    paymentstatus ENUM('Pending', 'Completed',  
'Failed'),  
    transactionid VARCHAR(100) UNIQUE,  
    FOREIGN KEY (orderid) REFERENCES `Order`(orderid)  
);
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
1 CREATE TABLE Payment (  
2     paymentid INT PRIMARY KEY,  
3     orderid INT,  
4     paymentmethod ENUM('Card', 'UPI', 'Wallet', 'NetBanking'),  
5     paymentstatus ENUM('Pending', 'Completed', 'Failed'),  
6     transactionid VARCHAR(100) UNIQUE,  
7     FOREIGN KEY (orderid) REFERENCES `Order`(orderid)  
8 );  
9  
10 describe Payment;
```

The bottom section displays the 'Result Grid' with the following data:

Field	Type	Null	Key	Default	Extra
paymentid	int	NO	PRI	NULL	
orderid	int	YES	MUL	NULL	
paymentmethod	enum('Card','UPI','Wallet','NetBanking')	YES		NULL	
paymentstatus	enum('Pending','Completed','Failed')	YES		NULL	
transactionid	varchar(100)	YES	UNI	NULL	

On the right side, there are icons for 'Result Grid', 'Form Editor', and a search icon.

9. Order_MenuItem table

Used to store information of orders places by customers

```
CREATE TABLE Order_MenuItem (  
    orderid INT,  
    itemid INT,  
    quantity INT,  
    PRIMARY KEY (orderid, itemid),  
    FOREIGN KEY (orderid) REFERENCES  
`Order`(orderid),  
    FOREIGN KEY (itemid) REFERENCES MenuItem(itemid)  
);
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
1 • CREATE TABLE Order_MenuItem (  
2     orderid INT,  
3     itemid INT,  
4     quantity INT,  
5     PRIMARY KEY (orderid, itemid),  
6     FOREIGN KEY (orderid) REFERENCES `Order`(orderid),  
7     FOREIGN KEY (itemid) REFERENCES MenuItem(itemid)  
8 );  
9  
10 • describe MenuItem;
```

Below the editor is a 'Result Grid' section. It includes a 'Filter Rows' search bar and an 'Export' button. The grid displays the following data:

Field	Type	Null	Key	Default	Extra
itemid	int	NO	PRI	NULL	
restaurantid	int	YES	MUL	NULL	
name	varchar(100)	YES		NULL	
price	decimal(10,2)	YES		NULL	
preparationtime	int	YES		NULL	
availability	tinyint(1)	YES		NULL	
category	varchar(100)	YES		NULL	

On the right side, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', and 'Field Types'.

Normalization (Applied in the solution proposed):

1. User (Already in 3NF)

- Each column is functionally dependent on the primary key (userid)
- There are no transitive dependencies

2. Restaurant (Already in 3NF)

- Each column is functionally dependent on the primary key (restaurantid)
- The foreign key “ownerid” references “userid” in the “User” table

3. MenuItem

- **1NF**: All attributes contain atomic values
- **2NF**: All non-key attributes are fully dependent on the primary key (itemid)
- **3NF**: No transitive dependencies

4. Order (LEFT IN DENORMALIZED STATE)

- Including “discountapplied” and “deliverypartnerid” in this table avoids the need for extra joins while fetching data related to an order's discount and delivery

5. Discount (Already in 3NF)

- Each column is functionally dependent on the primary key (discountid)
- No transitive dependencies

6. DeliveryPartner (Already in 3NF)

- Each column is functionally dependent on the primary key (deliverypartnerid)

7. Feedback (Already in 3NF)

- Each column is functionally dependent on the primary key (feedbackid)
- Foreign keys “userid” and “orderid” ensure referential integrity

8. Payment (Already in 3NF)

- Each column is functionally dependent on the primary key (paymentid)
- Foreign key “**ordered**” ensures referential integrity

9. Order_MenuItem (Bridge Table) (Already in 3NF)

- **Purpose:** Represents the M:N relationship between “**Order**” and “**MenuItem**”
- Each column is fully dependent on the composite primary key (orderid, itemid)

Denormalized Tables (Justification for keeping them as they are):

1. Order:

- Attributes like discountapplied and deliverypartnerid are included to simplify queries related to discounts and delivery partners
- Reducing joins improves performance for frequently accessed data

2. Order_MenuItem:

- Maintains normalization while efficiently handling the M:N relationship

Final glimpse of “**fooddelivery**”:

