1.0 Background of the Restaurant Chain

The restaurant franchise in question runs numerous sites throughout the region, serving a varied clientele. Each location seeks to deliver a distinctive culinary experience emphasizing quality and consistency. The franchise provides a diverse array of cuisines, encompassing appetizers, main meals, desserts, and beverages, according to the preferences of its target clientele.

Key Aspects:

- The network presently operates more than 10 outlets distributed throughout urban and suburban regions, hence enhancing accessibility for a wider audience.
- The restaurant serves families, working professionals, and culinary lovers desiring a varied cuisine and a pleasant dining atmosphere.
- Cuisine Offered: The menu features worldwide options including Italian, American, and Continental, alongside regionally inspired meals to cater to local tastes.
- Operational Difficulties: The rapidly expanding restaurant chain has multiple issues that require the establishment of a comprehensive Database Management System (DBMS):

1. Reservation Management: The manual processing of customer reservations frequently results in overbooking or missing bookings, leading to consumer discontent.
2. Inventory Monitoring: Erratic tracking of ingredient consumption and stock levels leads to wastage or the unavailability of essential ingredients.
3. Sales Reporting: The lack of automated reporting hinders the ability to monitor daily sales and assess the performance of menu items across locations.

The deployment of an integrated DBMS is critical for addressing operational inefficiencies, improving decision-making, and enhancing customer satisfaction through real-time insights. As Elmasri and Navathe (2020) highlight, enabling end-users to perform transactions directly, supported by robust and updatable databases, is essential for sectors like e-commerce and hospitality. Similarly, in the restaurant industry, digitizing activities such as reservations, inventory management, and sales reporting enables streamlined operations and better customer experiences.

1.2 Purpose of the Database Management System (DBMS)

The Database Management System (DBMS) for the restaurant chain is designed to tackle various operational difficulties and offer a consolidated platform for data management. The objective of the DBMS is as follows:

1. To Optimize Operations Across Locations

The restaurant franchise manages numerous locations, each with distinct operations. The DBMS will centralize functions including reservation management, inventory tracking, and sales monitoring across all branches, thus improving operational efficiency. Elmasri and Navathe (2020) assert that an effectively designed DBMS minimizes redundancy and enhances data accuracy through the integration of data sources.

2. To Improve Customer Experience via Effective Reservation and Inventory Management

The DBMS guarantees a flawless client experience with real-time reservation management and automated inventory adjustments. Avoiding overbookings and guaranteeing menu availability are essential for client happiness. Singh and Mittal (2021) emphasize that integrated systems substantially improve customer experience by reducing delays and errors.

3. Enhancing Data Reporting and Decision-Making

The DBMS facilitates the production of comprehensive reports, including daily sales summary, inventory status, and client trends. These insights facilitate data-driven decision-making, enabling the restaurant chain to respond adeptly to market demands.

1.3 Objectives of the Assessment

This evaluation seeks to build and implement a complete Database Management System that meets the operational and analytical requirements of the restaurant chain. The primary objectives are:

1. **To Design a Data Model That Is Stable**

The evaluation entails the development of an Entity-Relationship Diagram (ERD) that accurately depicts the relationships and dependencies within the restaurant's operations. The model guarantees scalability and clarity, thereby facilitating future improvements without the need for substantial redesign (Hoffer et al., 2020).

2. **To Guarantee Adherence to the Most Effective Methods of Database Normalization**

The design guarantees integrity and eliminates data redundancy by normalizing the database to the Third Normal Form (3NF). The assessment emphasizes the importance of establishing a clear and efficient database structure, as normalization is essential for ensuring data consistency.

3. **To Generate Reports That Fulfill the Organization's Operational Requirements**

The assessment entails formulating SQL queries to generate actionable results, including total sales by category, inventory status, and client reservation records. These reports correspond with the chain's strategic objectives of enhancing efficiency and profitability.

2. Requirements Analysis

2.1 Functional Requirements

1. **Manage customer reservations:**

- Customers may create, alter, and cancel reservations.
- Employees can authorize and monitor reservations.
- Reservations are monitored by branches and dates.

2. **Monitor inventory levels:**

- Inventory for each branch is assessed, encompassing stock quantities and replenishment needs.

- Collaboration with suppliers for inventory replenishment.

3. **Daily sales reports are generated:**

- Sales summaries for each branch are compiled on a daily basis.
- Revenue reports for restaurant categories and locations are produced.
- Analysis of sales by menu items and customer trends is conducted.

2.2 Non-Functional Requirements

1. **Scalability for future growth:**

- The system should support a growing number of restaurants, branches, and customers.
- Guarantee that the database can handle large volumes of transactions.

2. **Secure data handling:**

- Protect sensitive customer and payment information using encryption.
- Enforce strict user access rules for employees.

3. **High system availability:**

- Guarantee minimal downtime with a robust database infrastructure.
- Backup systems should be in place to restore data in the event of failures.

2.3 System Constraints

Financial constraints:

- The system must employ economical technologies.
- It is advised to use open-source database management systems like PostgreSQL or MySQL.

Existing hardware or software dependencies:

- The solution must be compatible with existing hardware or POS systems.
- The software must be compatible with the operating system and platform currently being used by the restaurant chain.

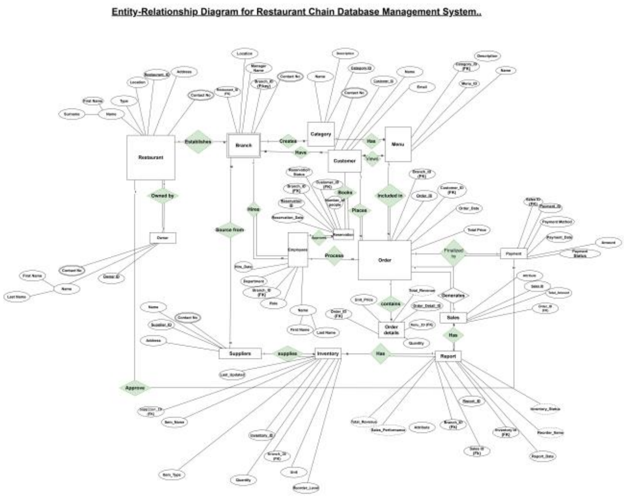
3.1 Data Model Overview

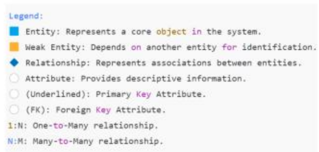
Entities	Attributes	Key Constraints	Other
Owner	Owner_ID, First_Name, Last_Name, Contact_No	Owner_ID (PK)	Validates Contact_No using a regex.
Restaurant	Restaurant_ID, Name, Type, Location, Owner_ID	Restaurant_ID (PK), Owner_ID (FK)	Cascading actions on Owner_ID
Branch	Branch_ID, Restaurant_ID, Location, Manager_Name, Contact_No	Branch_ID (PK), Restaurant_ID (FK)	Contact_No validated with a regex.
Employee	Employee_ID, Branch_ID, First_Name, Last_Name, Role, Hire_Date, Department	Employee_ID (PK), Branch_ID (FK)	
Customer	Customer_ID, First_Name, Last_Name, Email, Phone	Customer_ID (PK)	Email is unique.
Reservation	Reservation_ID, Customer_ID, Branch_ID, Reservation_Date, Number_of_People, Status	Reservation_ID (PK), Customer_ID (FK), Branch_ID (FK)	

Menu	Menu_ID, Item_Name, Description, Price, Category_ID	Menu_ID (PK), Category_ID (FK)	
Category	Category_ID, Category_Name, Category_Description	Category_ID (PK)	
Order	Order_ID, Customer_ID, Branch_ID, Order_Date, Total_Price	Order_ID (PK), Customer_ID (FK), Branch_ID (FK)	
Inventory	Inventory_ID, Ingredient_Name, Quantity, Unit, Reorder_Level, Branch_ID	Inventory_ID (PK), Branch_ID (FK)	
Supplier	Supplier_ID, Name, Address, Contact_No	Supplier_ID (PK)	
Report	Report_ID, Sales_Performance, Inventory_Status, Report_Date	Report_ID (PK)	

3. Database Design

3.2 Entity-Relationship Diagram (ERD)





Title: Entity-Relationship Diagram for Restaurant Chain Database Management System.

Description: The ERD illustrates the relationships between key entities, such as Restaurant, Branch, Employee, Customer, and Reservation, along with their attributes and constraints.

3.3 Description of Entities and Relationships

Entities and Attributes

1. Restaurant

Attributes:

- Restaurant: Unique identifier for the restaurant.
- Name: Name of the restaurant.
- Type: Category of food offered.

- Location: General location of the restaurant.

Relationships:

Owned by: Associates each restaurant to its owner (Owner_ID).

Establishes: Connects restaurants to their branches.

2. Branch

Attributes:

- Branch_ID: Distinct identification for the branch.
- Restaurant_ID: Associates the branch with a particular restaurant.
- Location: The physical address of the branch.
- Manager_Name: The name of the branch manager.
- Contact_No: The branch's contact number.

Relationships:

Has: Links the branch to reservations (Branch_ID in Reservation).

Employs: Links the branch to its employees (Branch_ID in Employee).

Supplies: Connects the branch to inventory items (Branch_ID in Inventory).

3. Owner

Attributes:

- Owner_ID: Unique identifier for the owner.
- First_Name: The Owner’s first name.
- Last_Name: The Owner’s last name.
- Contact_No: Contact number of the owner.

Relationships:

Owns: Links the owner to a restaurant.

4. Customer

Attributes:

- Customer_ID: Unique identifier for the customer.
- First_Name: The Customer’s first name.
- Last_Name: The Customer’s last name.
- Email: The Customer’s email address.
- Phone: The Customer’s phone number.

Relationships:

- **Books:** Links the customer to reservations.
- **Places:** Links the customer to orders.

5. Reservation

- **Reservation_ID:** Unique identifier for the reservation.
- **Customer_ID:** Associates the reservation with a customer.
- **Branch_ID:** Links the reservation to a branch.
- **Reservation_Date:** The Date of the reservation.
- **Number_of_People:** Number of people in the reservation.
- **Reservation_Status:** Status of the reservation (e.g., Confirmed, Pending).

Relationships:

Approved by: Links the reservation to an employee.

6. Employee

Attributes:

Employee_ID: Unique identifier for the employee.

Branch_ID: Links the employee to a branch.

First_Name: Employee’s first name.

Last_Name: Employee’s last name.

Role: Job role of the employee.

Hire_Date: Date the employee was hired.

Department: Department the employee works in.

Relationships:

- **Processes:** Links the employee to orders.
- **Approves:** Links the employee to reservations

7. Menu

Attributes:

Menu_ID: Unique identifier for the menu item.

Item_Name: Name of the menu item.

Description: Description of the menu item.

Price: The Price of the menu item.

Category_ID: Associates the menu item with a category.

Relationships:

Links the menu item to orders.

8. **Inventory**

Attributes:

Inventory_ID: Unique identifier for the inventory item.

Ingredient_Name: Name of the inventory item.

Quantity: Available Inventory.

Unit: A standard measurement (e.g., kg, liters).

Reorder_Level: The minimum threshold for initiating a reorder.

Branch_ID: Links inventory items to a branch.

Relationships: Supplied by: Links the inventory to suppliers.

9. **Summary of Relationships**

- Every reservation associates a customer with a branch.
- Every branch is affiliated with a restaurant.
- Employees oversee bookings and facilitate order processing for branches.
- Menu items are classified and associated with orders.
- Inventory is provided by suppliers and associated with branches.

3.4 Assumptions and Business Rules

- Each reservation is linked to only one customer.
- Each branch belongs to exactly one restaurant.
- Inventory items are managed independently for each branch.
- Each employee is assigned to only one branch at a time.
- A reservation can only be approved by one employee.
- Orders can include multiple menu items.
- Ingredients in the inventory are used exclusively for menu items.
- Each restaurant must have at least one branch.
- Customers can book multiple reservations.
- Employees must belong to a branch to process orders or approve reservations.
- Categories group menu items based on cuisine or dish type.
- Suppliers provide inventory for multiple branches.
- Daily sales are recorded for each branch individually.
- Reports are generated for inventory and sales separately.
- Menu items are linked to a single category.
- Payment details for orders are stored securely.
- Customers can update or cancel reservations before the reservation date.
- Low-stock inventory triggers reorder notifications.
- Each menu item has a unique price and description.
- Restaurants must have an assigned owner.

4.0 Normalization Process

4.1 Initial ERD Analysis

The preliminary ERD design encompassed the fundamental linkages and qualities necessary for the restaurant management system. Nonetheless, it displayed the subsequent problems:

1. Data Redundancy:

Certain characteristics, like Category_Name in the Menu object, are redundantly duplicated for each menu item. This redundancy would result in heightened storage demands and discrepancies during updates.

Attributes connected to inventory, including Ingredient_Name and Branch_ID, exhibited a lack of normalization, which may result in data duplication for ingredients utilized across many branches.

2. Partial Dependencies:

In the Menu table, elements such as Category_Name were functionally dependent on Menu_ID but could be segregated into a separate table to prevent redundancy.

In the Inventory database, properties such as Branch_ID resulted in partial dependencies, as inventory may be associated with numerous branches.

2. **Transitive Dependencies:**

    Certain attributes in specific tables relied on non-primary keys. For instance, Sales_Performance in the Reports table was indirectly related to other attributes and relied on Sales_ID.

4. Complex Relationships:
The relationships between Employee processing Orders and approving Reservations were not clearly delineated, resulting in potential inconsistencies in role allocations.

Requirement for Normalization:

Normalization was routinely implemented to address these issues:

- Eliminated redundancy by partitioning attributes into associated tables.
- Guaranteed that all non-key attributes were entirely dependent on their primary key.
- Eliminated transitive dependencies by establishing supplementary entities, including Category and Supplier.
- Streamlined relationships by the explicit delineation of foreign keys and the enforcement of integrity restrictions.

4.2 Normalization Process for All Entities (1NF, 2NF, 3NF)

This report contains an in-depth explanation of the normalization process for all entities in your ERD, as well as populated tables with 5 to 10 rows at each stage (1NF, 2NF, and 3NF).

Entity 1: Restaurant

Unnormalized Restaurant Table

Name	Type	Location	Owner_Name
Pizza Palace	Italian	London, New York	John Doe
Burger Haven	American	Chicago, Dallas	Jane Smith
Sushi World	Japanese	Tokyo, Osaka	Hiro Tanaka
Tandoor Hut	Indian	Delhi, Mumbai	Raj Patel
Cafe Delight	French	Paris, Lyon	Marie Dupont

Step 1: First Normal Form (1NF)

Rule: Ensure atomic values and eliminate multivalued attributes.

Issues:

1. **Location contains multiple values.**
2. **Owner_Name is a composite attribute.**

Normalized Restaurant Table (1NF):

Restaurant_ID	Name	Type	Location	Owner_First_Name	Owner_Last_Name
1	Pizza Palace	Italian	London	John	Doe
1	Pizza Palace	Italian	New York	John	Doe
2	Burger Haven	American	Chicago	Jane	Smith
2	Burger Haven	American	Dallas	Jane	Smith
3	Sushi World	Japanese	Tokyo	Hiro	Tanaka
3	Sushi World	Japanese	Osaka	Hiro	Tanaka
4	Tandoor Hut	Indian	Delhi	Raj	Patel
4	Tandoor Hut	Indian	Mumbai	Raj	Patel
5	Cafe Delight	French	Paris	Marie	Dupont
5	Cafe Delight	French	Lyon	Marie	Dupont

Step 2: Second Normal Form (2NF)

Rule: Eliminate partial dependencies.

Issues:

The Owner_First_Name and Owner_Last_Name are solely dependent on the Restaurant_ID, not on the complete composite key (Restaurant_ID, Location).

Solution: The table is divided into distinct Restaurant and Location tables.

Restaurant Table (2NF):

Restaurant_ID	Name	Type	Owner_First_Name	Owner_Last_Name
1	Pizza Palace	Italian	John	Doe
2	Burger Haven	American	Jane	Smith
3	Sushi World	Japanese	Hiro	Tanaka
4	Tandoor Hut	Indian	Raj	Patel
5	Cafe Delight	French	Marie	Dupont

Location Table (2NF):

Restaurant_ID	Location
1	London
1	New York
2	Chicago
2	Dallas
3	Tokyo
3	Osaka
4	Delhi
4	Mumbai
5	Paris
5	Lyon

Step 3: Third Normal Form (3NF)

Rule: Eliminate transitive dependencies.

Issues:

In order to prevent redundancy, the Owner_First_Name and Owner_Last_Name columns may be transferred to an Owner table.

Solution: Utilize a foreign key to refer to a distinct Owner table.

Restaurant Table (3NF):

Restaurant_ID	Name	Type	Owner_ID
1	Pizza Palace	Italian	101
2	Burger Haven	American	102
3	Sushi World	Japanese	103
4	Tandoor Hut	Indian	104
5	Cafe Delight	French	105

Owner Table (3NF):

Owner_ID	First_Name	Last_Name
101	John	Doe
102	Jane	Smith
103	Hiro	Tanaka
104	Raj	Patel
105	Marie	Dupont

Location Table (3NF):

Restaurant_ID	Location
1	London
1	New York
2	Chicago
2	Dallas
3	Tokyo
3	Osaka
4	Delhi
4	Mumbai
5	Paris
5	Lyon

Entity 2: Branch

Unnormalized Branch Table

Branch_ID	Restaurant_ID	Location	Manager_Name	Contact_No
101	1	London	Alice Johnson	1234567890
102	1	New York	Bob Smith	0987654321
103	2	Chicago	Clara Adams	1122334455
104	2	Dallas	Daniel Brown	2233445566
105	3	Tokyo	Emi Tanaka	3344556677

Step 1: First Normal Form (1NF)

Rule: Guarantee that the values are atomic.

Issues:

All values are already atomic; no changes needed.

Normalized Branch Table (1NF):

Branch_ID	Restaurant_ID	Location	Manager_Name	Contact_No
101	1	London	Alice Johnson	1234567890
102	1	New York	Bob Smith	0987654321
103	2	Chicago	Clara Adams	1122334455
104	2	Dallas	Daniel Brown	2233445566
105	3	Tokyo	Emi Tanaka	3344556677

Step 2: Second Normal Form (2NF)

Rule: Eliminate partial dependencies.

Problems:

Location, Manager_Name, and Contact_No are all non-key attributes that are contingent upon the entire principal key (Branch_ID). 2NF is already satisfied by the table.

Step 3: Third Normal Form (3NF)

Rule: Discard transitive dependencies.

Issues:

No transitive dependencies. Already, the table meets the requirements of 3NF.

Final Branch Table (3NF):

Branch_ID	Restaurant_ID	Location	Manager_Name	Contact_No
101	1	London	Alice Johnson	1234567890
102	1	New York	Bob Smith	0987654321
103	2	Chicago	Clara Adams	1122334455
104	2	Dallas	Daniel Brown	2233445566
105	3	Tokyo	Emi Tanaka	3344556677

Entity 3: Customer

Unnormalized Customer Table

Step 1: First Normal Form (1NF)

Rule: Guarantee that the values are atomic.

Issues:

All attributes contain atomic values; no multivalued attributes exist.

Normalized Customer Table (1NF):

Customer_ID	First_Name	Last_Name	Email	Phone
1	John	Doe	john.doe@gmail.com	1234567890
2	Jane	Smith	jane.smith@gmail.com	0987654321
3	Alice	Brown	alice.brown@gmail.com	1122334455
4	Bob	Johnson	bob.johnson@gmail.com	2233445566
5	Emma	Wilson	emma.wilson@gmail.com	3344556677
6	Liam	Davis	liam.davis@gmail.com	4455667788
7	Olivia	Miller	olivia.miller@gmail.com	5566778899
8	Noah	Moore	noah.moore@gmail.com	6677889900
9	Ava	Taylor	ava.taylor@gmail.com	7788990011
10	Mason	Anderson	mason.anderson@gmail.com	8899001122

Step 2: Second Normal Form (2NF)

Rule: Eliminate partial dependencies.

Problems:

All attributes are entirely contingent upon the fundamental key (Customer_ID). 2NF is satisfied by the data.

Normalized Customer Table (1NF):

Customer_ID	First_Name	Last_Name	Email	Phone
1	John	Doe	john.doe@gmail.com	1234567890
2	Jane	Smith	jane.smith@gmail.com	0987654321
3	Alice	Brown	alice.brown@gmail.com	1122334455
4	Bob	Johnson	bob.johnson@gmail.com	2233445566
5	Emma	Wilson	emma.wilson@gmail.com	3344556677
6	Liam	Davis	liam.davis@gmail.com	4455667788
7	Olivia	Miller	olivia.miller@gmail.com	5566778899
8	Noah	Moore	noah.moore@gmail.com	6677889900
9	Ava	Taylor	ava.taylor@gmail.com	7788990011
10	Mason	Anderson	mason.anderson@gmail.com	8899001122

Step 3: Third Normal Form (3NF)

Rule: Eliminate transitive dependencies.

Final Customer Table (3NF):

Customer_ID	First_Name	Last_Name	Email	Phone
1	John	Doe	john.doe@gmail.com	1234567890
2	Jane	Smith	jane.smith@gmail.com	0987654321
3	Alice	Brown	alice.brown@gmail.com	1122334455
4	Bob	Johnson	bob.johnson@gmail.com	2233445566
5	Emma	Wilson	emma.wilson@gmail.com	3344556677
6	Liam	Davis	liam.davis@gmail.com	4455667788
7	Olivia	Miller	olivia.miller@gmail.com	5566778899
8	Noah	Moore	noah.moore@gmail.com	6677889900
9	Ava	Taylor	ava.taylor@gmail.com	7788990011
10	Mason	Anderson	mason.anderson@gmail.com	8899001122

Problems:

There are no transitive dependencies. The 3NF is satisfied by the structure in its current state.

Entity 4: Menu

Unnormalized Menu Table

Menu_ID	Item_Name	Description	Price	Category_Name
1	Margherita	Classic pizza	8.99	Pizza
2	Cheeseburger	Grilled beef patty	10.49	Burgers
3	Caesar Salad	Crisp romaine	6.99	Salads
4	Espresso	Italian coffee shot	2.99	Beverages
5	Spaghetti	Classic pasta	12.99	Pasta
6	Chocolate Cake	Rich chocolate	5.99	Desserts
7	Latte	Coffee with milk	4.49	Beverages
8	BBQ Chicken	Chicken in BBQ sauce	14.99	Mains

First Normal Form (1NF): Step 1

Rule: Guarantee that the values are atomic.

Problems:

Atomic values are present in all attributes; there are no multivalued attributes.

Normalized Menu Table (1NF):

Menu_ID	Item_Name	Description	Price	Category_Name
1	Margherita	Classic pizza	8.99	Pizza
2	Cheeseburger	Grilled beef patty	10.49	Burgers
3	Caesar Salad	Crisp romaine	6.99	Salads
4	Espresso	Italian coffee shot	2.99	Beverages
5	Spaghetti	Classic pasta	12.99	Pasta
6	Chocolate Cake	Rich chocolate	5.99	Desserts
7	Latte	Coffee with milk	4.49	Beverages
8	BBQ Chicken	Chicken in BBQ sauce	14.99	Mains

Step 2: Second Normal Form (2NF)

Rule: Eliminate partial dependencies.

Problems:

Menu_ID is a determinant of Category_Name; however, it may be relocated to a distinct Category table.

Solution: Create a Category table and reference it in the Menu table using a foreign key.

Menu Table (2NF):

Menu_ID	Item_Name	Description	Price	Category_ID
1	Margherita	Classic pizza	8.99	101
2	Cheeseburger	Grilled beef patty	10.49	102
3	Caesar Salad	Crisp romaine	6.99	103
4	Espresso	Italian coffee shot	2.99	104
5	Spaghetti	Classic pasta	12.99	105
6	Chocolate Cake	Rich chocolate	5.99	106
7	Latte	Coffee with milk	4.49	104
8	BBQ Chicken	Chicken in BBQ sauce	14.99	107

Category Table (2NF):

Category_ID	Category_Name	Description
101	Pizza	Classic pizzas
102	Burgers	Grilled sandwiches
103	Salads	Healthy salads
104	Beverages	Hot and cold drinks
105	Pasta	Italian pastas
106	Desserts	Sweet treats
107	Mains	Main course dishes

Step 3: Third Normal Form (3NF)

Rule: Eliminate transitive dependencies.

Problems:

The Menu and Category tables do not contain any transitive dependencies.

Final Menu Table (3NF):

Menu_ID	Item_Name	Description	Price	Category_ID
1	Margherita	Classic pizza	8.99	101
2	Cheeseburger	Grilled beef patty	10.49	102
3	Caesar Salad	Crisp romaine	6.99	103
4	Espresso	Italian coffee shot	2.99	104
5	Spaghetti	Classic pasta	12.99	105
6	Chocolate Cake	Rich chocolate	5.99	106
7	Latte	Coffee with milk	4.49	104
8	BBQ Chicken	Chicken in BBQ sauce	14.99	107

Final Category Table (3NF):

Category_ID	Category_Name	Description
101	Pizza	Classic pizzas
102	Burgers	Grilled sandwiches
103	Salads	Healthy salads
104	Beverages	Hot and cold drinks
105	Pasta	Italian pastas
106	Desserts	Sweet treats
107	Mains	Main course dishes

Entity 7: Inventory

Unnormalized Inventory Table

Inventory_ID	Ingredient_Name	Quantity	Unit	Reorder_Level	Branch_ID
1	Tomato	50	kg	10	101
2	Cheese	20	kg	5	102
3	Lettuce	30	kg	8	103
4	Coffee Beans	15	kg	3	104
5	Pasta	40	kg	12	105
6	Chocolate	25	kg	7	106
7	Chicken	60	kg	15	107

First Normal Form (1NF): Step 1

Rule: Guarantee that the values are atomic.

Problems:

Atomic values are present in all attributes; there are no multivalued attributes.

Normalized Inventory Table (1NF):

Inventory_ID	Ingredient_Name	Quantity	Unit	Reorder_Level	Branch_ID
1	Tomato	50	kg	10	101
2	Cheese	20	kg	5	102
3	Lettuce	30	kg	8	103
4	Coffee Beans	15	kg	3	104
5	Pasta	40	kg	12	105
6	Chocolate	25	kg	7	106
7	Chicken	60	kg	15	107

Step 2: Second Normal Form (2NF)

Rule: Eliminate partial dependencies.

Problems:

The fundamental key (Inventory_ID) is the determining factor for all non-key attributes. 2NF is satisfied by the data.

Step 3: Third Normal Form (3NF)

Rule: Eliminate transitive dependencies.

Problems:

The table does not contain any transitive dependencies. The 3NF is satisfied by the structure in its current state.

Final Inventory Table (3NF):

Inventory_ID	Ingredient_Name	Quantity	Unit	Reorder_Level	Branch_ID
1	Tomato	50	kg	10	101
2	Cheese	20	kg	5	102
3	Lettuce	30	kg	8	103
4	Coffee Beans	15	kg	3	104
5	Pasta	40	kg	12	105
6	Chocolate	25	kg	7	106
7	Chicken	60	kg	15	107

Entity: Reservation Unnormalized Reservation Table

Reservation_ID	Customer_ID	Branch_ID	Reservation_Date	Number_of_People	Reservation_Status
1	101	1	2025-01-01	4	Confirmed
2	102	2	2025-01-02	2	Pending
3	103	1	2025-01-03	6	Confirmed
4	104	3	2025-01-04	3	Cancelled
5	105	4	2025-01-05	5	Confirmed
6	106	1	2025-01-06	2	Confirmed
7	107	2	2025-01-07	8	Pending
8	108	3	2025-01-08	4	Confirmed
9	109	4	2025-01-09	7	Confirmed
10	110	1	2025-01-10	2	Cancelled

First Normal Form (1NF): Step 1

Rule: Guarantee that the values are atomic. Eliminate attributes that have multiple values.

Normalized Reservation Table (1NF):

Step 2: Second Normal Form (2NF)

Rule: Eliminate partial dependencies.

Analysis:

All non-key attributes (Reservation_Date, Number_of_People, Reservation_Status) depend wholly on the primary key (Reservation_ID).

No further adjustments are required as the table already satisfies 2NF.

Step 3: Third Normal Form (3NF)

Rule: Eliminate transitive dependencies.

Examination:

There are no transitive dependencies. 3NF is satisfied by the data.

Final Reservation Table (3NF):

Reservation_ID	Customer_ID	Branch_ID	Reservation_Date	Number_of_People	Reservation_Status
1	101	1	2025-01-01	4	Confirmed
2	102	2	2025-01-02	2	Pending
3	103	1	2025-01-03	6	Confirmed
4	104	3	2025-01-04	3	Cancelled
5	105	4	2025-01-05	5	Confirmed
6	106	1	2025-01-06	2	Confirmed
7	107	2	2025-01-07	8	Pending
8	108	3	2025-01-08	4	Confirmed
9	109	4	2025-01-09	7	Confirmed
10	110	1	2025-01-10	2	Cancelled

Entity: Sales

Unnormalized Sales Table

Sales_ID	Order_ID	Sales_Date	Total_Amount
1	201	2025-01-01	500.00
2	202	2025-01-02	200.00
3	203	2025-01-03	300.00
4	204	2025-01-04	450.00
5	205	2025-01-05	600.00
6	206	2025-01-06	700.00
7	207	2025-01-07	800.00
8	208	2025-01-08	1000.00
9	209	2025-01-09	250.00
10	210	2025-01-10	300.00

First Normal Form (1NF): Step 1

Rule: Guarantee that the values are atomic.

Normalized Sales Table (1NF):

Normalized Sales Table (1NF):

Sales_ID	Order_ID	Sales_Date	Total_Amount
1	201	2025-01-01	500.00
2	202	2025-01-02	200.00
3	203	2025-01-03	300.00
4	204	2025-01-04	450.00
5	205	2025-01-05	600.00
6	206	2025-01-06	700.00
7	207	2025-01-07	800.00
8	208	2025-01-08	1000.00
9	209	2025-01-09	250.00
10	210	2025-01-10	300.00

Step 2: Second Normal Form (2NF)

Rule: Eliminate partial dependencies.

Analysis:

All attributes that are not key are entirely dependent on the primary key (Sales_ID).

The table already meets 2NF, so no additional modifications are necessary.

Step 3: Third Normal Form (3NF)

Rule: Eliminate transitive dependencies.

Examination:

There are no transitive dependencies. 3NF is satisfied by the data.

Final Sales Table (3NF):

Sales_ID	Order_ID	Sales_Date	Total_Amount
1	201	2025-01-01	500.00
2	202	2025-01-02	200.00
3	203	2025-01-03	300.00
4	204	2025-01-04	450.00
5	205	2025-01-05	600.00
6	206	2025-01-06	700.00
7	207	2025-01-07	800.00
8	208	2025-01-08	1000.00
9	209	2025-01-09	250.00
10	210	2025-01-10	300.00

Entity: Report

Unnormalized Report Table

Report_ID	Sales_Performance	Inventory_Status	Report_Date
1	High	Sufficient	2025-01-01
2	Medium	Low	2025-01-02
3	Low	Critical	2025-01-03
4	High	Sufficient	2025-01-04
5	Medium	Low	2025-01-05
6	High	Sufficient	2025-01-06
7	Low	Critical	2025-01-07
8	High	Sufficient	2025-01-08
9	Medium	Low	2025-01-09
10	Low	Critical	2025-01-10

First Normal Form (1NF): Step 1

Rule: Guarantee that the values are atomic.

Normalized Report Table (1NF):

Normalized Report Table (1NF):

Report_ID	Sales_Performance	Inventory_Status	Report_Date
1	High	Sufficient	2025-01-01
2	Medium	Low	2025-01-02
3	Low	Critical	2025-01-03
4	High	Sufficient	2025-01-04
5	Medium	Low	2025-01-05
6	High	Sufficient	2025-01-06
7	Low	Critical	2025-01-07
8	High	Sufficient	2025-01-08
9	Medium	Low	2025-01-09
10	Low	Critical	2025-01-10

Step 2: Second Normal Form (2NF)

Rule: Eliminate partial dependence.

Examination:

All non-key properties are entirely dependent on the primary key (Report_ID).

No other modifications are necessary, as the table already complies with Second Normal Form (2NF).

Step 3: Third Normal Form (3NF)

Rule: Eliminate transitive dependencies.

Examination:

There are no transitive dependencies present. The table complies with Third Normal Form (3NF).

Final Report Table (3NF):

Report_ID	Sales_Performance	Inventory_Status	Report_Date
1	High	Sufficient	2025-01-01
2	Medium	Low	2025-01-02
3	Low	Critical	2025-01-03
4	High	Sufficient	2025-01-04
5	Medium	Low	2025-01-05
6	High	Sufficient	2025-01-06
7	Low	Critical	2025-01-07
8	High	Sufficient	2025-01-08
9	Medium	Low	2025-01-09
10	Low	Critical	2025-01-10

4.4 Final Normalized Design in Third Normal Form (3NF)

The final design includes a collection of normalized tables wherein:

1. Tables for Restaurants and Branches:

- The Restaurant table is connected to Branches via Restaurant_ID (FK).
- Every branch is distinctly defined by features including Location, Manager_Name, and Contact_No.

2. Tables for Customers and Reservations:

- Customers are distinctly identifiable by Customer_ID, guaranteeing the accurate storage of all customer data, including Email and Phone.
- Reservations are standardized, associating Customer_ID and Branch_ID to prevent redundancy of branch or customer information.

3. Tables for Menu and Categories:

The Menu table references the Category table via Category_ID (FK), so preventing the repetition of category names in the menu items.

4. Tables for Orders, Sales, and Reports:

- Orders are executed inside a standardized framework, utilizing Customer_ID and Menu_ID while avoiding the redundancy of customer or menu information.
- Sales transactions are associated with orders, facilitating precise revenue tracking while distinguishing sales information from the order entity.
- Reports consolidate information from Sales and Inventory tables, offering performance insights while avoiding unnecessary data storage.

Elimination of Transitive Dependencies:

1. **Within the Menu entity:**

The Category_Name has been relocated to a distinct Category table, associated by Category_ID.

2. **Within the Reports entity:**

Properties such as Sales_Performance and Inventory_Status were directly linked to Report_ID, eliminating any indirect dependencies via Sales_ID or other properties.

Throughout all entities:

Attributes were meticulously examined and categorized into corresponding tables, guaranteeing that each non-key attribute was entirely and directly reliant on its primary key.

Advantages of the Final Structure:

- Enhanced Data Integrity: By eliminating redundancies and dependencies, updates or deletions are guaranteed to only affect pertinent tables, thereby preserving consistency.
- Decreased Storage Requirements: By dividing attributes into smaller tables, duplication is minimised, and storage usage is optimized.
- Improved Query Performance: Efficient joins and queries are facilitated by well-structured tables, particularly for intricate reporting requirements.

- Scalability: The system can effortlessly accommodate new branches, menu categories, or reports without the need for substantial schema modifications.

This finalized 3NF structure guarantees that the database is optimized for efficient data management and retrieval, thereby satisfying the functional and non-functional requirements specified in the assessment brief.

Calculating Reorder Level for Inventory:

The Reorder Level in the inventory table is determined by past consumption trends and replenishment lead time. The calculation can be performed using the formula:

Formula for Reorder Level:

Illustrative Computation:

Mean Daily Consumption: 10 units/day

Delivery Time: 5 days

Safety Stock: 20 units

Reorder Threshold:

This value is stored as a static threshold in the inventory database for the purpose of monitoring stock levels.

SQL Query for Branch Revenue:

To calculate the revenue generated by each branch, the following SQL query can be used:

```sql
SELECT
    Branch_ID,
    SUM(Total_Amount) AS Total_Revenue
FROM
    Sales
JOIN
    Orders ON Sales.Order_ID = Orders.Order_ID
GROUP BY
    Branch_ID;
```

Explanation:

The query computes the total revenue (SUM(Total_Amount)) for each branch (Branch_ID) by combining the Sales table with the Orders table.

The GROUP BY clause categorizes the revenue totals by branch.

This finalized 3NF structure ensures the database is optimized for effective data management and retrieval, therefore fulfilling the functional and non-functional objectives outlined in the assessment brief.

5.1 Create Database

The **CREATE DATABASE** statement is commonly used in SQL to initialise a new database. As demonstrated in the script provided, the command for creating the database is:

```sql
CREATE DATABASE RestaurantChainDB;
```

This corresponds with the elucidation offered by W3Schools (n.d.), which emphasizes the role of this statement in establishing a database within a SQL context.

5.2 The **USE** statement in SQL is utilised to designate a certain database for subsequent actions. The command to utilize the database in the provided script is:

```sql
USE RestaurantChainDB;
```

This aligns with W3Schools' (n.d.) description, which characterizes the USE statement as a method for designating the database context for running SQL queries.

5.3 **SQL Script for Creating Tables with Key Constraints**

Customer Table

Result Grid | Filter Rows: | Edit:

Customer_ID	First_Name	Last_Name	Email
1	Emma	Johnson	emma.johnson@example.com
2	Liam	Smith	liam.smith@example.com
3	Olivia	Brown	olivia.brown@example.com
4	Noah	Davis	noah.davis@example.com
5	Sophia	Taylor	sophia.taylor@example.com
6	James	Wilson	james.wilson@example.com
7	Mia	Clark	mia.clark@example.com
8	Lucas	Moore	lucas.moore@example.com
9	Charlotte	Hall	charlotte.hall@example.com
10	Elijah	Lee	elijah.lee@example.com
NULL	NULL	NULL	NULL

Customer 17

The Customer table covers ten records, each of which contains critical customer information, including Customer_ID, First_Name, Last_Name, Email, and Phone. The totality and accuracy of critical customer data are guaranteed by the enforcement of NOT NULL constraints on all mandatory fields. The insertion of incomplete records is prohibited by the presence of these constraints, which is essential for the integrity of the database.

Although all mandatory fields have been completed, certain fields, such as Phone, may accept NULL values to account for instances in which the data is not immediately accessible. In order to guarantee that data entry remains adaptable while adhering to database standards, this method is advised (W3Schools, n.d.).

Branch Table

Result Grid | Filter Rows: | Edit: | Export/Import:

Branch_ID	Restaurant_ID	Location	Manager_Name	Contact_No
1	1	London	Alice Johnson	1234567890
2	2	New York	Bob Smith	0987654321
3	3	Tokyo	Hiro Tanaka	9876543210
4	4	Mexico City	Carlos Garcia	4567891230
5	5	Rome	Luigi Rossi	1237894560
6	6	Texas	James Carter	6543219870
7	7	Mumbai	Rajesh Kumar	7891234560
8	8	Beijing	Li Wei	3219876540
9	9	Berlin	Anna Schmidt	9873216540
10	10	Sydney	Jack Wilson	6547891230
NULL	NULL	NULL	NULL	NULL

Branch 18

The Branch table contains information about ten branches, including the following columns: Branch_ID, Restaurant_ID, Location, Manager_Name, and Contact_No. The Restaurant_ID foreign key is the connecting factor between each branch and its corresponding restaurant in this structure. In order to guarantee the data's integrity and dependability, NOT NULL constraints are implemented for all mandatory elements, including Branch_ID, Restaurant_ID, and Location.

In order to accommodate situations in which the data may not be immediately accessible, the optional elements, including Manager_Name and Contact_No, permit NULL values. This design guarantees adaptability without sacrificing the table's functionality or purpose. These constraints are consistent with the most effective methods of relational database administration, which promotes accuracy and consistency (W3Schools, no date).

Result Grid | Filter Rows: | Edit: | Export/Import:

Category_ID	Category_Name	Description
1	Main Course	Meals as mains
2	Snacks	Light snacks
3	Desserts	Sweet treats
4	Beverages	Drinks and refreshments
5	Appetizers	Starters and small bites
6	Soups	Warm and comforting soups
7	Salads	Healthy and fresh salads
8	Grilled Items	Grilled specialties
9	Sandwiches	Quick meals
10	Breakfast	Morning meals
NULL	NULL	NULL

Category 19

The Category table categorizes menu items into ten predetermined categories, such as Main Course, Snacks, and Desserts. The principal key is the Category_ID column, which uniquely identifies each category. Details regarding each category are furnished by the supplementary columns, Category_Name and Description.

In order to guarantee data integrity, constraints are enforced, including the NOT NULL constraint on Category_ID and Category_Name, which guarantees that each category is designated a distinct identifier and name (W3Schools, no date). In order to accommodate categories that may not necessitate further information, the Description column permits NULL values. This table design facilitates efficient data retrieval and promotes scalability when connected to the Menu table through foreign key relationships.

Menu Table

Menu_ID	Name	Description	Price	Category_ID
1	Margherita Pizza	Classic cheese and tomato pizza	9.99	1
2	Cheeseburger	Beef patty with melted cheese	8.49	2
3	Chocolate Cake	Rich chocolate dessert	6.99	3
4	Mojito	Mint and lime refreshing drink	4.99	4
5	Nachos	Crispy nachos with cheese	5.99	5
6	Tomato Soup	Warm tomato soup with herbs	4.49	6
7	Caesar Salad	Classic Caesar salad with dressing	7.99	7
8	Grilled Chicken	Succulent grilled chicken	12.99	8
9	Club Sandwich	Double-layered sandwich	6.49	9
10	Pancakes	Fluffy pancakes with syrup	5.99	10
NULL	NULL	NULL	NULL	NULL

The Menu table specifies the possibilities inside a restaurant system, with each menu item clearly identified by the Menu_ID, which serves as the main key. The table includes essential attributes such as Name, Description, Price, and Category_ID. The Price column is regulated with a CHECK constraint to prohibit the input of negative numbers.

The Category_ID serves as a foreign key, linking the menu item to its respective category in the Category table, thus improving the organization and retrieval of menu items by category. Columns such as Name and Price are subject to NOT NULL constraints, ensuring that every menu item has a name and a price. Nonetheless, the Description column accommodates NULL values, permitting the omission of details. This comprehensive structure guarantees data integrity and scalability for the menu system. Moreover, the integrity checker verifies actions that alter the database, assuring compliance with constraints and database regulations (Connolly and Begg, 2015).

Restaurant Table

Result Grid | Filter Rows: | Edit: | Export/Import:

Restaurant_ID	Name	Type	Owner_ID
1	Pizza Palace	Italian	1
2	Burger Haven	American	2
3	Sushi Corner	Japanese	3
4	Taco Fiesta	Mexican	4
5	Pasta Paradise	Italian	5
6	BBQ Grill	Barbecue	6
7	Curry Kitchen	Indian	7
8	Dim Sum Delight	Chinese	8
9	Vegan Bliss	Vegan	9
10	Seafood Shack	Seafood	10
NULL	NULL	NULL	NULL

Restaurant 22 ×

The Restaurant table is an essential element of the database, used to store and handle data on different eateries. Every restaurant is distinctly identified by the Restaurant_ID, which functions as the primary key. Essential properties like Name, Type, and Owner_ID guarantee thorough data representation. The Type column classifies restaurants by cuisine, following established categories such as "Italian," "American," and "Japanese," so ensuring data consistency.

The Owner_ID serves as a foreign key, connecting the restaurants to the relevant entries in the Owner table, thus preserving referential integrity. This relational framework, adhering to normalization principles, guarantees that the data is suitably organized to prevent redundancy and improve scalability. Columns like as Name and Type are subject to NOT NULL constraints, ensuring that each restaurant entry possesses established fundamental features (Connolly and Begg, 2015).

Supplier Table

Supplier_ID	Name	Address	Contact_No
1	Fresh Produce Co.	123 Green Lane, London	1234567890
2	Meat Supplies Ltd.	456 Red Street, New York	0987654321
3	Dairy Farms	789 Blue Avenue, Tokyo	9876543210
4	Grain Wholesalers	101 Yellow Road, Mexico City	4567891230
5	Baking Essentials	202 Pink Blvd, Rome	1237894560
6	Sweeteners Inc.	303 Orange Dr, Texas	6543219870
7	Egg Producers	404 Violet Lane, Mumbai	7891234560
8	Leafy Greens	505 Indigo Street, Beijing	3219876540
9	Chocolate World	606 Brown Rd, Berlin	9873216540
10	Pork Distributors	707 Black Alley, Sydney	6547891230
NULL	NULL	NULL	NULL

Supplier 23 ×

Report on Supplier Table

The **Supplier** table is a vital element of the relational database, storing details about suppliers that provide inventory items to the restaurant branches. Each supplier is uniquely identified by the Supplier_ID, serving as the primary key. Key attributes such as Name, Address, and Contact_No ensure comprehensive representation of supplier details, facilitating effective management and communication.

The Name and Contact_No columns are governed by NOT NULL constraints, ensuring that essential information about each supplier is provided. However, the Address column permits NULL values, offering flexibility for cases where address details may not be immediately available. This table structure adheres to the principles of normalization, as it ensures data integrity, reduces redundancy, and enhances scalability within the database system (Connolly and Begg, 2015).

Owner Table

Result Grid | Filter Rows: | Edit:

Owner_ID	First_Name	Last_Name
1	John	Doe
2	Alice	Smith
3	Robert	Brown
4	Emma	Jones
5	Liam	Taylor
6	Sophia	Wilson
7	James	Davis
8	Mia	Clark
9	Oliver	Moore
10	Amelia	Hall
NULL	NULL	NULL

Owner 24

The **Owner** table contains essential information regarding the proprietors overseeing the restaurants within the system. Every owner is distinctly identified by the Owner_ID, which functions as the primary key. The table comprises properties like First_Name and Last_Name, facilitating the precise identification of each owner.

The First_Name and Last_Name columns are subject to NOT NULL restrictions, mandating the inclusion of crucial owner details. This approach improves data precision and mitigates the occurrence of incomplete information in the database. The lack of other optional features in this table enhances simplicity and concentrates exclusively on identifying owners.

This table complies with normalization standards by guaranteeing the uniqueness of each record and the atomicity of characteristics, hence preserving data integrity and eradicating redundancy (Connolly and Begg, 2015).

Customer Table

Customer_ID	First_Name	Last_Name	Email	Phone
1	Emma	Johnson	emma.johnson@example.com	1234567890
2	Liam	Smith	liam.smith@example.com	0987654321
3	Olivia	Brown	olivia.brown@example.com	9876543210
4	Noah	Davis	noah.davis@example.com	4567891230
5	Sophia	Taylor	sophia.taylor@example.com	1237894560
6	James	Wilson	james.wilson@example.com	6543219870
7	Mia	Clark	mia.clark@example.com	7891234560
8	Lucas	Moore	lucas.moore@example.com	3219876540
9	Charlotte	Hall	charlotte.hall@example.com	9873216540
10	Elijah	Lee	elijah.lee@example.com	6547891230
NULL	NULL	NULL	NULL	NULL

Customer 25

The Customer table offers a detailed summary of customer data, uniquely identifying each customer using the Customer_ID, which functions as the primary key. This guarantees that each consumer in the system is uniquely represented.

The principal attributes in this table comprise:

First_Name and Last_Name: These fields are constrained by NOT NULL requirements to guarantee that each customer is properly identified by their whole name.
Email: This field is essential for communication and is subject to a NOT NULL and UNIQUE constraint to exclude duplicate or absent email records, hence facilitating efficient customer identification.
Phone: Although not obligatory (NULL values are permissible), this field offers further contact information for clients.

The architecture complies with normalization principles by preserving atomic properties and eliminating redundancy. This guarantees that the table facilitates effective data storage and retrieval (Connolly and Begg, 2015).

Inventory Table

Inventory_ID	Ingredient_Name	Quantity	Unit	Reorder_Level	Branch_ID
1	Cheese	50	kg	10	1
2	Tomatoes	30	kg	5	2
3	Beef	20	kg	5	3
4	Chicken	40	kg	8	4
5	Flour	100	kg	15	5
6	Sugar	60	kg	10	6
7	Eggs	200	pcs	50	7
8	Lettuce	[illegible]	kg	6	8
9	Chocolate	15	kg	3	9
10	Bacon	20	kg	5	10

The Inventory table is essential for controlling the restaurant's supply and guaranteeing operational efficiency by monitoring ingredients and their quantities. Every inventory item is distinctly recognized by the Inventory_ID, which functions as the primary key.

The table encompasses essential attributes:

Ingredient_Name: Denotes the designation of the ingredient, crucial for the identification and management of inventory items.

Quantity: Denotes the present stock level of each element. A CHECK constraint guarantees the entry of only non-negative values, hence maintaining data integrity.

Unit: Specifies the measurement unit (e.g., kg, pcs), ensuring uniformity in inventory management.

Reorder_Level: Indicates the minimum inventory threshold that activates a reorder. This prevents stock shortages, which is essential for seamless restaurant operations.

Branch_ID: Serves as a foreign key that associates each inventory item with a particular branch, facilitating localized inventory monitoring and administration.

This table configuration guarantees precise and dynamic inventory oversight. It facilitates real-time updates of stock levels when inventory is utilized or restocked, a crucial practice for sustaining operational efficiency (Elmasri and Navathe, 2016).

Result Grid Filter Rows: Edit: Export/Import:

Reservation_ID	Customer_ID	Branch_ID	Reservation_Date	No_of_People	Status
1	1	1	2025-01-01	4	Confirmed
2	2	2	2025-01-02	2	Pending
3	3	3	2025-01-03	5	Confirmed
4	4	4	2025-01-04	3	Cancelled
5	5	5	2025-01-05	6	Confirmed
6	6	6	2025-01-06	7	Confirmed
7	7	7	2025-01-07	8	Pending
8	8	8	2025-01-08	9	Confirmed
9	9	9	2025-01-09	4	Confirmed
10	10	10	2025-01-10	2	Cancelled
NULL	NULL	NULL	NULL	NULL	NULL

Reservation 27 ×