App-Based Restaurant Ordering Database System

Group Members:

Sidhant Kumar
14657868

Praneeth reddi
14602680

Ashikul Kabir
14676807

Rakshit Madan
14661346

Atharv Vinay Deo
14681284

Course:
MIS 632 – Database Analysis and Design for Business

Instructor:
Dr. Qizhi Dai

Term:
Spring 2024

Date of Submission:
June 10, 2024

Table of Contents

Questions Covered:

1. Description of the business operation task for the database design. Develop the requirements for the database design. (10 points)

2. According to the above requirements, design the entity-relationship model. The ER model should include four to six entities. (10 points)

3. Given the ER model, design the relational schema. (10 points)

4. Describe the process for creating the database, including the specification of tables, fields in tables, primary keys, foreign keys, indexes. (10 points)

5. Identify three business questions that can be answered with the data in the database; Write SQL statements to retrieve the data to answer these questions. At least one SQL statement should use data from multiple tables. At least one SQL statement should use GROUP BY clause. (10 points)

Business Operation Task:

The objective of this project is to develop a comprehensive database system to manage app based online ordering. This system encompasses multiple processes, including customer interactions, order processing, and the fulfilment of orders through pickup or delivery. The goal of the Database Management System (DBMS) is to enhance these operations by providing a robust tool for effective data management, accurate record-keeping, and seamless integration among different components of the system.

To achieve this, the system must enable smooth customer interactions, allowing customers to place orders easily through a mobile app or website. It should ensure efficient order processing, with real-time updates on order status, and support a seamless flow from the kitchen to delivery or pickup. The database must securely store sensitive information. These key functionalities aim to improve operational efficiency and customer satisfaction in the online ordering process.

Description of the Business Operation Task:

App based Online ordering is a complex process which involves several key steps which are:

1. **Customer Interaction:**

- Customers use a mobile app or website to access the online ordering system.
- Customers browse the menu, select their items, and finalize their orders.
- Customers choose between pickup and delivery and provide payment details.

2. **Order Processing:**

- Once the order is placed, the system handles it, ensuring correct allocation to the designated restaurant location.
- Order details are conveyed to the kitchen staff for preparation.

3. **Delivery and Pickup:**

- Following processing, the order is either assigned to a delivery driver or prepared for customer pickup.
- Throughout this process, the system continuously monitors the order status, providing real-time updates to the customer.

Requirements for the Database Design:

To support this business operation, our database needs to meet the following criteria:

1. **Customer Management:**

- Track client information, such as name, address, and phone number.
- Ensure secure storage of sensitive data, including encrypted credit card information and personal details.
- Enable tracking of order history and preferences through customer profiles.

2. **Menu Management:**

- Maintain an extensive menu catalogue that includes the name, description, cost, and category of each item.
- Permit item modification (e.g., altering ingredients).
- Monitor the availability of items and manage connections with orders.

3. **Order Processing:**

- Track every order's detail, including the date, customer's ID, restaurant's ID, order status, and total amount.
- Support unique identifiers to link relevant entities such as patrons, restaurants, and order items.
- Provide real-time updates on order status to keep customers informed.

4. **Restaurant Locations:**

- Record details about every restaurant location, such as address, phone number, and hours of operation.
- Manage relationships with both employees and orders to facilitate order processing and assignment.

5. **Delivery and Pickup:**

- Track delivery or pickup information, including delivery address, assigned delivery person, and estimated time of delivery.
- Ensure prompt order fulfilment by tracking delivery status.
- Provide real-time updates to customers on their order status.

6. Employee Management:

- Track staff details, such as name, position, and designated dining area.
- Manage the assignment of delivery drivers and kitchen staff to orders.
- Support employee roles and responsibilities within the system.

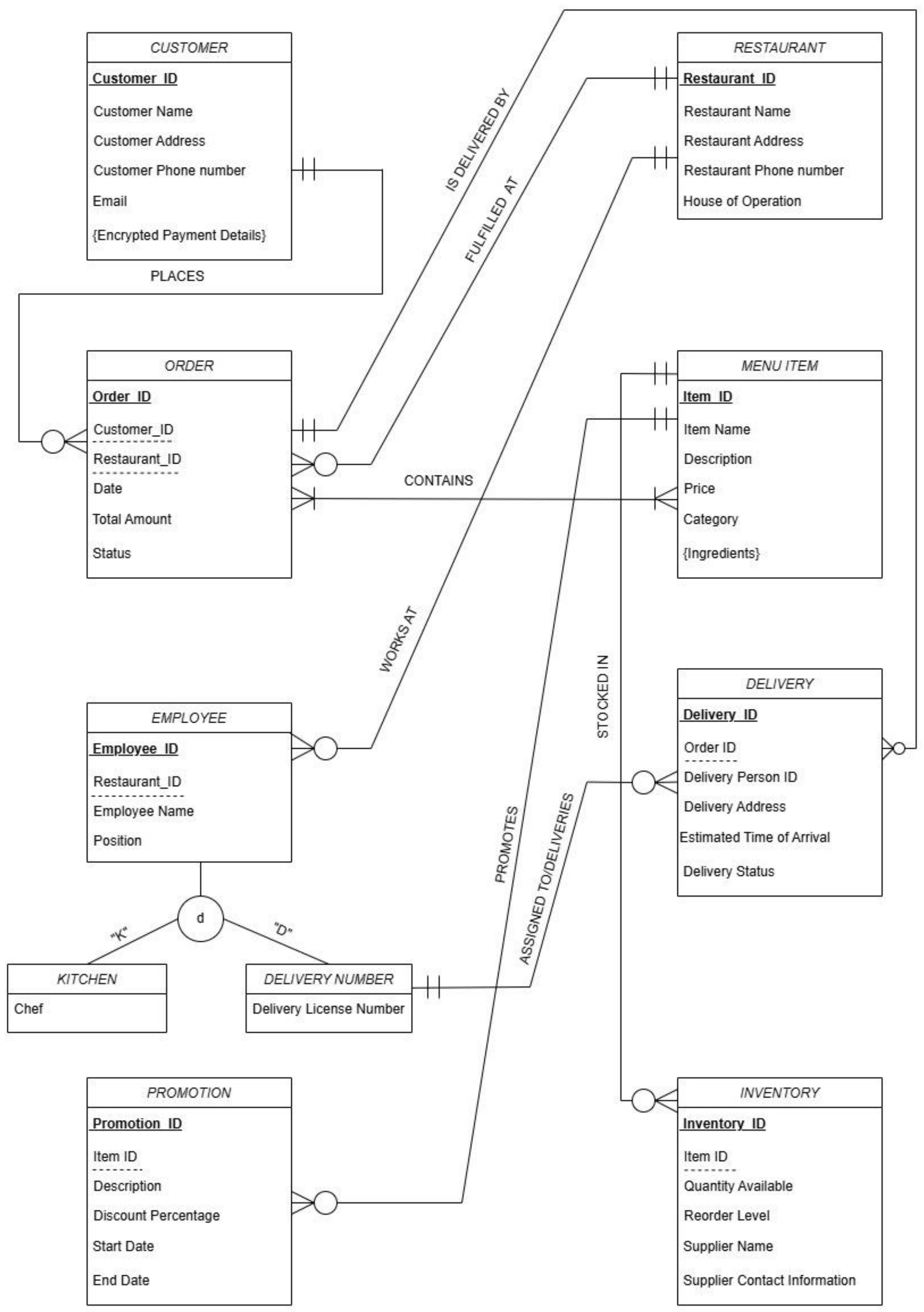
7. Inventory Management:

- Monitor stock levels of ingredients and supplies in real-time to ensure the kitchen is well-stocked.
- Set thresholds for automatic reordering of ingredients when stock levels fall below a certain point.
- Maintain records of suppliers, including contact information and past order histories.

8. Marketing and Promotions Management:

- Track details of current and past promotions, discounts, and special offers.
- Enable segmentation of customers based on their order history and preferences to tailor marketing campaigns.
- Analyse the effectiveness of marketing campaigns by tracking redemption rates and sales uplift during promotional periods.

ER Model for App-Based Restaurant Ordering System



DBMS ERD Blueprint: Online Food Ordering and Delivery System

The entity-relationship diagram (ERD) for this project is meticulously designed to streamline the operations of online food ordering and delivery, similar to services like DoorDash or Uber Eats. This ERD precisely addresses the business requirements by establishing a systematic structure to enhance customer interactions, manage orders efficiently, and seamlessly integrate with external delivery services.

By modelling the interactions between customers, the restaurant, and third-party delivery services, the ERD provides a robust framework to handle the complexities of food ordering and delivery in a digital landscape. It ensures secure and efficient data handling, which is critical for maintaining customer trust and managing sensitive information. The primary aim is to foster a reliable and efficient environment that not only meets the immediate needs of food delivery but also enhances customer satisfaction through timely and accurate service delivery. This design supports the restaurant's goal of extending its reach beyond traditional dine-in experiences to a broader digital audience, thereby optimizing both customer engagement and operational workflow.

ERD Explanation:

Customer Entity: Represents the patrons who engage with the online food ordering platform, holding key contact details and secure payment information. This entity is essential for tailoring the user experience, supporting both regular customers and newcomers alike.

Customer ID (Primary Key)
Customer Name
Customer Address (Composite of street, city, state, zip code)
Customer Phone Number
Email
Encrypted Payment Details (Multivalued to accommodate multiple payment methods)

Order Entity: Acts as the central record for each transaction, capturing details about customer orders, including their origin, processing, and final status.

Order ID (Primary Key)
Date
Total Amount
Status
Customer ID (Foreign Key referencing Customer)
Restaurant ID (Foreign Key referencing Restaurant)

Menu Item Entity: Catalogues the array of offerings available, with attributes that allow for detailed descriptions and pricing. This entity supports menu customization and reflects availability, crucial for accurate order processing.

Item ID (Primary Key)
Item Name
Description
Price
Category
Ingredients (Multivalued to list all ingredients)

Restaurant Entity: Holds information on each restaurant location, becoming a central point that ties together orders, delivery, and employees. It forms the backbone of the operational database, facilitating order allocation and preparation.

Restaurant ID (Primary Key)
Restaurant Name
Restaurant Address (Composite)
Restaurant Phone Number
Hours of Operation

Delivery Entity: Tracks the delivery aspect of orders, assigning drivers and managing delivery schedules. It ensures that customers are informed of their order's journey from the kitchen to their doorstep.

Delivery ID (Primary Key)
Order ID (Foreign Key referencing Order)
Delivery Person ID (Foreign Key referencing Employee)
Delivery Address
Estimated Time of Arrival
Delivery Status

Employee Entity: Categorized into kitchen staff and delivery drivers, this entity tracks all personnel involved in the food preparation and delivery process, facilitating role assignment and operational efficiency.

Employee ID (Primary Key)
Employee Name
Position
Restaurant ID (Foreign Key referencing Restaurant)

Inventory Entity: Manages the stock of ingredients and supplies required for menu items. This entity helps in tracking the quantity available, reordering needs, and supplier details.

Inventory ID (Primary Key)
Item ID (Foreign Key referencing Menu Item)
Quantity Available
Reorder Level
Supplier Name
Supplier Contact Information

Promotion Entity: Manages details of promotions and discounts offered on menu items. This entity helps in marketing and boosting sales through targeted promotions.

Promotion ID (Primary Key)
Item ID (Foreign Key referencing Menu Item)
Description
Discount Percentage
Start Date
End Date

Relationships:

Places: Connects Customer to Order, depicting that a customer can place none, one, or multiple orders. Each Order can be linked to one customer.

Cardinality: Optional Many to Mandatory One

Fulfilled At: Links Order to Restaurant, indicating that each order is processed at a single restaurant location. A restaurant can handle multiple orders.

Cardinality: Optional Many to Mandatory One

Contains: Represents a many-to-many relationship between Orders and Menu Items, allowing for detailed order customization and tracking. Each Order can contain multiple menu items, and each menu item can belong to multiple orders.

Cardinality: Many to Many

Works At: Connects Employee to Restaurant, showing where each employee is based, which is essential for organizing staff and streamlining operations. Each employee works at one restaurant. A restaurant can hire many employees.

Cardinality: Optional Many to Mandatory One

Assigned To/Delivers & Is Delivered By: Links Delivery to Order and Delivery Driver, securing the connection between the order, its delivery, and the responsible driver.

Cardinality: Optional Many to Mandatory One (Driver to Delivery) and Mandatory One to Optional One (Delivery to Order)

Stocked In: Links Inventory to Menu Item, indicating that each inventory item is associated with one menu item, and each menu item can be associated with multiple inventory records.

Cardinality: Optional Many to Mandatory One

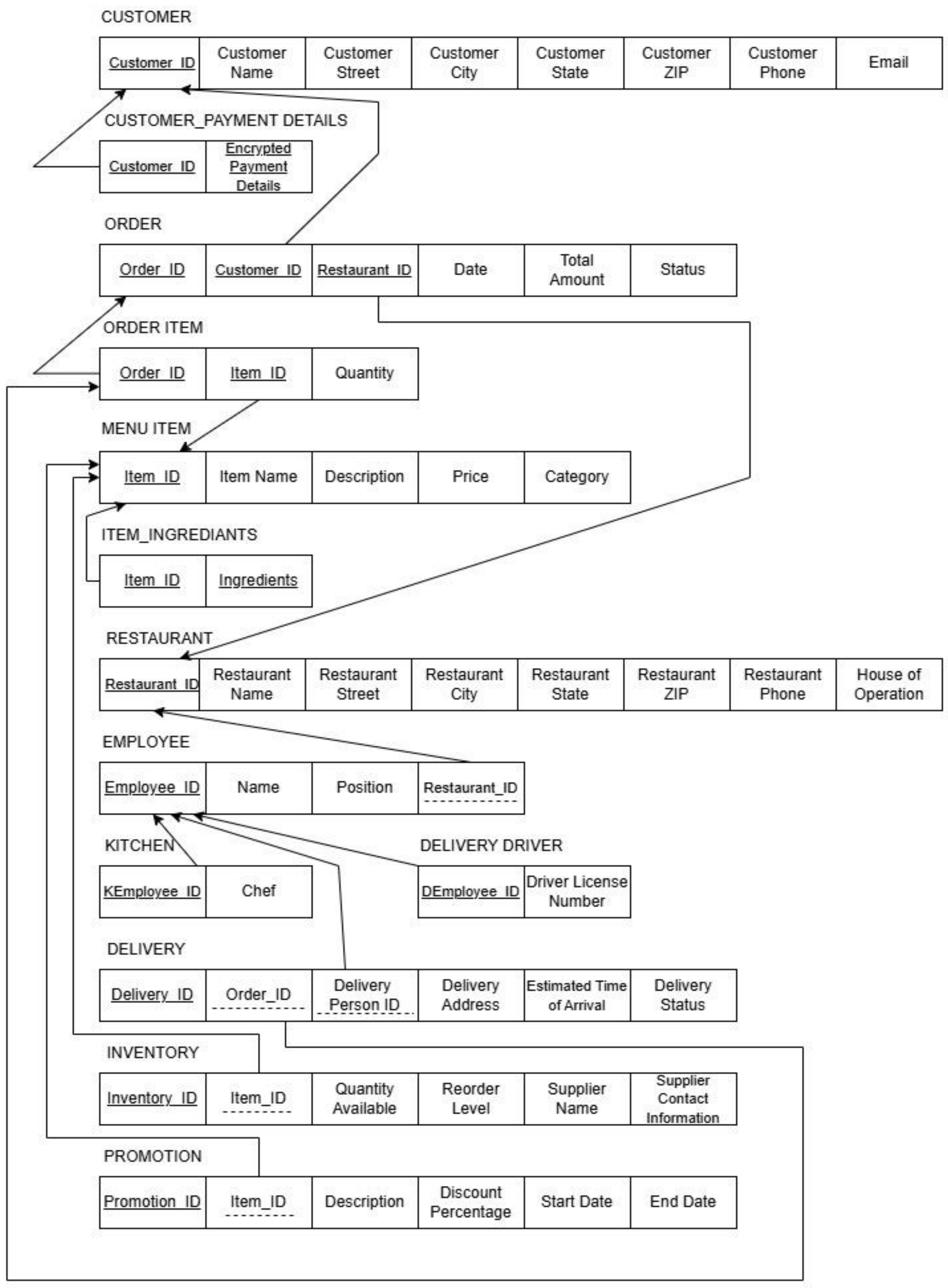
Promotes: Links Promotion to Menu Item, indicating that each promotion is associated with one or more menu items. Each menu item can have multiple promotions.

Cardinality: Optional Many to Mandatory One

Keys:

Primary Keys are used to uniquely identify each entity.
Foreign Keys create necessary links between entities, supporting relational database integrity and facilitating complex queries.

Relational Schema for App based Ordering System


CUSTOMER
Customer_ID | Customer Name | Customer Street | Customer City | Customer State | Customer ZIP | Customer Phone | Email
CUSTOMER_PAYMENT DETAILS
Customer_ID | Encrypted Payment Details
ORDER
Order_ID | Customer_ID | Restaurant_ID | Date | Total Amount | Status
ORDER ITEM
Order_ID | Item_ID | Quantity
MENU ITEM
Item_ID | Item Name | Description | Price | Category
ITEM_INGREDIANTS
Item_ID | Ingredients
RESTAURANT
Restaurant_ID | Restaurant Name | Restaurant Street | Restaurant City | Restaurant State | Restaurant ZIP | Restaurant Phone | House of Operation
EMPLOYEE
Employee_ID | Name | Position | Restaurant_ID
KITCHEN
KEmployee_ID | Chef
DELIVERY DRIVER
DEmployee_ID | Driver License Number
DELIVERY
Delivery_ID | Order_ID | Delivery Person ID | Delivery Address | Estimated Time of Arrival | Delivery Status
INVENTORY
Inventory_ID | Item_ID | Quantity Available | Reorder Level | Supplier Name | Supplier Contact Information
PROMOTION
Promotion_ID | Item_ID | Description | Discount Percentage | Start Date | End Date

Normalization to 3NF of the Schema

The schema designed for the app-based ordering system is structured to meet the criteria for normalization up to the third normal form (3NF), ensuring data integrity and minimizing redundancy.

First Normal Form (1NF):

Principle: Each table has a primary key, and all attributes contain atomic values. There are no repeating groups or arrays.

Application:

Each table in the schema has a clearly defined primary key.
Attributes such as Customer Street, Customer City, Customer State, and Customer Zip in the Customer table are atomic and non-repeating.
The separation of Ingredients into its own table from Menu Item ensures no repeating groups.

Second Normal Form (2NF):

Principle: Each table's non-key attributes are fully functionally dependent on the entire primary key.

Application:

In the Order Item table, the attribute Quantity is fully dependent on the composite primary key (Order ID, Item ID), ensuring that no non-key attribute is partially dependent on a part of the primary key.
Other tables such as Order, Menu Item, and Restaurant also exhibit this principle, where all non-key attributes are fully dependent on the primary key.

Third Normal Form (3NF):

Principle: There are no transitive dependencies; all attributes are only dependent on the primary key within their table.

Application:

In the Customer table, attributes such as Customer Street, Customer City, Customer State, and Customer Zip are directly dependent on Customer ID and not on any other attribute, preventing redundancy.

The separation of sensitive payment information into the Customer Payment Details table ensures that sensitive data is decoupled from general customer information, adhering to security best practices and normalization principles. Tables like Order, Delivery, and Employee ensure that non-key attributes are non-transitively dependent on their respective primary keys.

Conclusion:

The schema designed for the app-based ordering system is carefully normalized to the third normal form (3NF). Each table adheres to the principles of 1NF, 2NF, and 3NF, ensuring that:

All tables have primary keys and atomic attributes (1NF). Non-key attributes are fully functionally dependent on the entire primary key (2NF). There are no transitive dependencies, and all attributes are only dependent on the primary key (3NF). This normalization approach ensures data integrity, minimizes redundancy, and supports efficient database management for the online ordering system.

Details of the Schema:

CUSTOMER:

Customer ID: Unique identifier for a customer.
Customer Name: Full name of the customer.
Customer Street, Customer City, Customer State, Customer Zip: Address components.
Customer Phone: Phone number.
Email: Email address.

CUSTOMER PAYMENT DETAILS:

Customer ID: Foreign Key to CUSTOMER.
Encrypted Payment Details: Stores payment information securely.

ORDER:

Order ID: Unique identifier for an order.
Customer ID: Foreign Key to CUSTOMER.
Restaurant ID: Foreign Key to RESTAURANT.
Date: Date when the order was placed.
Total Amount: The financial sum of the order.
Status: The status of the order (e.g., pending, completed).

ORDER ITEM:

Order ID: Foreign Key to ORDER, part of composite PK.
Item ID: Foreign Key to MENU_ITEM, part of composite PK.
Quantity: Number of each menu item ordered.

MENU ITEM:

Item ID: Unique identifier for a menu item.
Item Name: Name of the menu item.
Description: Description of the item.
Price: Cost of the item.
Category: Categorization of the menu item.

ITEM_INGREDIENTS:

Item ID: Foreign Key to MENU_ITEM.
Ingredients: List of ingredients used in the menu item.

RESTAURANT:

Restaurant ID: Unique identifier for a restaurant location.
Restaurant Name: Name of the restaurant.
Restaurant Street, Restaurant City, Restaurant State, Restaurant Zip: Address components.
Restaurant Phone: Contact number.
Hours of Operation: Operating hours of the restaurant.

EMPLOYEE:

Employee ID: Unique identifier for an employee.
Name: Full name of the employee.
Position: Job title or position.
Restaurant ID: Foreign Key to RESTAURANT.

KITCHEN:

KEmployeeID: Foreign Key to EMPLOYEE, for kitchen staff.
Chef: Employee preparing orders

DELIVERY_DRIVER:

DEmployeeID: Foreign Key to EMPLOYEE, for delivery drivers.
Driver License Number: License number of the delivery driver.

DELIVERY:

Delivery ID: Unique identifier for a delivery instance.
Order ID: Foreign Key to ORDER.
DEmployee ID: Foreign Key to EMPLOYEE.
Delivery Address: Address where the order is to be delivered.
Estimated Time of Arrival: Expected delivery time.
Delivery Status: Status of the delivery.

INVENTORY:

Inventory ID: Unique identifier for an inventory record.
Item ID: Foreign Key to MENU_ITEM.
Quantity Available: Number of units available in stock.
Reorder Level: The threshold level at which the item should be reordered.
Supplier Name: Name of the supplier providing the inventory item.
Supplier Contact Information: Contact details of the supplier.

PROMOTION:

Promotion ID: Unique identifier for a promotion.
Item ID: Foreign Key to MENU_ITEM.
Description: Details about the promotion or discount.
Discount Percentage: The percentage discount offered by the promotion.
Start Date: Date when the promotion starts.
End Date: Date when the promotion ends.

Database Implementation Strategy: Creating the Database

Introduction:

This section outlines the systematic process involved in building the database for an app-based online ordering system. It details the specific steps required to convert the conceptual data model into an operational relational database. These procedures are essential to ensure that the operational implementation aligns perfectly with the theoretical design, enabling efficient data management and optimized query performance.

Overview of Database Functionality:

A critical component of the app-based online ordering system's smooth operation is the database. It serves as the foundation for managing various types of data, including customer information, order histories, menu management, delivery tracking, and employee scheduling. Every element of the database is designed to support the following features:

Customer Management:

The database stores a broad range of data about each customer, including contact information, order history, and payment options.
This enables personalized service and effective customer interaction, enhancing the overall user experience.

Order Processing:

By maintaining thorough records of every order, the database ensures close synchronization between the kitchen and delivery systems.
It allows for real-time order tracking and status updates, ensuring timely and accurate processing of orders.

Inventory and Menu Management:

The database keeps an up-to-date catalogue of every menu item, including prices, availability, and descriptions.
This supports effective inventory control and allows for menu modifications based on customer preferences and stock levels.

Delivery Logistics:

Delivery information is meticulously tracked from dispatch to delivery, improving delivery efficiency.
Customers receive timely updates, enhancing satisfaction with the service.

Employee Scheduling and Management:

The database manages employee details, assisting with staffing needs and scheduling across multiple restaurant locations.
This ensures that the right number of staff is available to handle orders and deliveries efficiently.

Promotion Management:

The database manages details of promotions and discounts offered on menu items.
It helps in marketing efforts by tracking the effectiveness of promotions and enabling targeted campaigns based on customer order history.

Step 1: Specification of Tables and Fields

Overview:

A fundamental step in creating a database is converting the Entity-Relationship (ER) model into actual tables. To convert the entities and relationships shown in the ER diagram into tables that the database management system can store and manage, they must be mapped. Every entity becomes a table, and every attribute inside the table becomes a field. The relational integrity of the database is maintained by relationships in the ER model, which determine how tables relate to one another through foreign keys.

Table Specifications:

The complete details for every table generated from the ER model are listed below. These include the table name, purpose, fields along with their respective data types, and a synopsis of each field. This specification makes sure that every table efficiently collects and arranges the data according to the ER model's design.

Customer Table:

Name: CUSTOMER
Purpose: Stores all relevant information about customers who use the online ordering system.
Description: This table reflects the 'Customer' entity in the ER model, capturing essential contact information required to process orders and communicate with the customer.

Fields:

- **Customer_ID (INT, PK):** A unique identifier for each customer.
- **Customer_Name (VARCHAR (100)):** The customer's full name.
- **Customer_Street (VARCHAR (255)):** The customer's street address.
- **Customer_City (VARCHAR (50)):** The city where the customer resides.
- **Customer_State (VARCHAR (50)):** The state where the customer resides.
- **Customer_Zip (VARCHAR (10)):** The postal code of the customer's address.
- **Customer_Phone (VARCHAR (15)):** The customer's telephone number.
- **Email (VARCHAR (100)):** The customer's email address.

Order Table:

Name: Order
Purpose: Manages information about customer orders from initiation to delivery.
Description: Corresponds to the 'Order' entity in the ER diagram, including its relationship to customers and restaurants, and manages the lifecycle of each order.

Fields:

- **OrderID (INT, PK):** A unique identifier for each order.
- **CustomerID (INT, FK):** Links to the Customer table, identifying who placed the order.
- **RestaurantID (INT, FK):** Identifies the restaurant from which the order is fulfilled.
- **OrderDate (DATE):** The date on which the order was placed.
- **TotalAmount (DECIMAL (10,2)):** The total cost of the order.
- **Status (VARCHAR (50)):** The status of the order (e.g., pending, completed).

Menu Item Table:

Name: MENU_ITEM
Purpose: Contains details about the items available for order.
Description: This table is derived from the 'Menu Item' entity in the ER model, providing a catalogue of what the restaurant offers.
Fields:

- **ItemID (INT, PK):** A unique identifier for each menu item.
- **Name (VARCHAR (100)):** The name of the menu item.
- **Description (TEXT):** A description of the menu item.
- **Price (DECIMAL (10,2)):** The price of the menu item.
- **Category (VARCHAR (50)):** The category to which the menu item belongs (e.g., appetizer, entree).

Restaurant Table:

Name: RESTAURANT
Purpose: Maintains details about each restaurant location within the area.
Description: This table aligns with the 'Restaurant' entity in the ER model, capturing essential information about the physical locations from which the restaurant operates, facilitating order fulfilment and local management.

Fields:

- **Restaurant_ID (INT, PK):** A unique identifier for each restaurant location.
- **Restaurant_Name (VARCHAR (100)):** The name of the restaurant.
- **Restaurant_Street (VARCHAR (255)):** The street address of the restaurant.
- **Restaurant_City (VARCHAR (50)):** The city where the restaurant is located.
- **Restaurant_State (VARCHAR (50)):** The state where the restaurant is located.
- **Restaurant_Zip (VARCHAR (10)):** The postal code of the restaurant's address.
- **Restaurant_Phone (VARCHAR (15)):** The contact number for the restaurant.
- **Hours (VARCHAR (50)):** Operating hours of the restaurant.

Employee Table:

Name: EMPLOYEE
Purpose: Records information about all employees, including their roles and assignments.
Description: This table represents the 'Employee' entity, detailing staff members across different locations, and supports managing their roles in connection with the operational needs of the restaurant.

Fields:
- **Employee ID (INT, PK):** A unique identifier for each employee.
- **Name (VARCHAR (100)):** The full name of the employee.
- **Position (VARCHAR (50)):** The role or job title of the employee (e.g., waiter, chef, manager).
- **RestaurantID (INT, FK):** Links to the Restaurant table, denoting the restaurant where the employee works.

Delivery Table:

Name: DELIVERY
Purpose: Tracks the progress and details of order deliveries.
Description: This table is derived from the 'Delivery' entity in the ER model and is crucial for managing the logistics of order delivery, ensuring timely and accurate service to customers.

Fields:
- **DeliveryID (INT, PK):** A unique identifier for each delivery.
- **OrderID (INT, FK):** Connects to the Order table, specifying which order is being delivered.
- **DeliveryPersonID (INT, FK):** References the Employee table, identifying the delivery person.
- **Address (VARCHAR (255)):** The delivery address for the order.
- **EstimatedTimeOfArrival (DATETIME):** The expected time of arrival for the delivery.
- **Status (VARCHAR (50)):** The status of the delivery (e.g., en route, delivered).

Order Item Table:

Name: ORDER_ITEM
Purpose: Manages the items within each order, especially useful in many-to-many relationships between orders and menu items.

Description: This associative table implements the many-to-many relationship between Order and Menu Item entities, capturing detailed item-level information for each order.

Fields:

- **OrderID (INT, FK):** Links to the Order table, associating it with a specific order.
- **ItemID (INT, FK):** Links to the Menu Item table, identifying the ordered item.
- **Quantity (INT):** The number of units of the item ordered.

Kitchen Staff Table:

Name: KITCHEN
Purpose: Details specific information about kitchen staff, distinct from other employees.
Description: This table focuses on the specialization and operational roles of kitchen staff within the restaurant, essential for managing culinary operations and menu execution.

Fields:

- **KEmployeeID (INT, FK):** Links to the Employee table, identifying kitchen staff members.
- **Specialization (VARCHAR (100)):** Describes the specific role or culinary expertise of the kitchen staff.

Delivery Driver Table:

Name: DELIVERY_DRIVER
Purpose: Manages information specific to employees designated as delivery drivers.
Description: This table is crucial for managing the details of employees who are specifically engaged in the delivery of orders, ensuring compliance and efficiency in delivery operations.

Fields:

- **DEmployeeID (INT, FK):** References the Employee table, identifying drivers.
- **LicenseNumber (VARCHAR (50)):** The driver's license number, necessary for legal and identification purposes.

Customer Payment Details Table:

Name: CUSTOMER_PAYMENT_DETAILS
Purpose: Stores encrypted payment details for customers.
Description: This table securely manages sensitive payment information, crucial for processing transactions and ensuring customer trust and privacy.

Fields:

- **CustomerID (INT, FK):** Links to the Customer table.
- **PaymentDetails (VARCHAR (255)):** Encrypted data representing the customer's payment methods.

Item Ingredients Table:

Name: ITEM_INGREDIENTS
Purpose: Manages the ingredients for each menu item, supporting inventory and recipe details.
Description: This table provides detailed information about the ingredients of each menu item, aiding in inventory management and compliance with food safety regulations.

Fields:

- **IngredientID (INT, PK):** A unique identifier for each ingredient.
- **ItemID (INT, FK):** Links to the Menu Item table, identifying which menu item the ingredient belongs to.
- **Name (VARCHAR (100)):** The name of the ingredient.
- **Quantity (VARCHAR (50)):** The quantity of the ingredient used for the menu item.

Inventory Table:

Name: INVENTORY
Purpose: Manages the stock of ingredients and supplies required for menu items.
Description: This table provides critical data for inventory management, ensuring the kitchen is well-stocked and able to meet customer demand.

Fields:

- **InventoryID (INT, PK):** A unique identifier for each inventory record.
- **ItemID (INT, FK):** Links to the Menu_Item table.
- **QuantityAvailable (INT):** Number of units available in stock.

- **ReorderLevel (INT):** The threshold level at which the item should be reordered.
- **SupplierName (VARCHAR (100)):** Name of the supplier providing the inventory item.
- **SupplierContactInformation (VARCHAR (255)):** Contact details of the supplier.

Promotion Table:

Name: PROMOTION
Purpose: Manages details of promotions and discounts offered on menu items.
Description: This table helps in managing promotions and discounts, enhancing marketing efforts, and boosting sales.

Fields:

- **PromotionID (INT, PK):** A unique identifier for each promotion.
- **ItemID (INT, FK):** Links to the Menu_Item table.
- **Description (TEXT):** Details about the promotion or discount.
- **DiscountPercentage (DECIMAL (5, 2)):** The percentage discount offered by the promotion.
- **StartDate (DATE):** Date when the promotion starts.
- **EndDate (DATE):** Date when the promotion ends.

In connection with the ER Model. Every table that is made corresponds exactly with an entity in the ER model, maintaining the relationships and integrity that were specified in the conceptual design. Fields match each entity's attributes, guaranteeing thorough data collection. To create relationships between tables, foreign keys are used, which replicate the connections shown in the ER diagram. By carefully mapping the intended functionality and relationships within the system, this mapping guarantees that the physical database accurately reflects the conceptual design.

Step 2: Defining Primary and Foreign Keys:

Overview:

Primary keys (PK) and foreign keys (FK) are crucial components of relational databases that preserve data integrity and specify table relationships. Primary keys ensure that every record in a table is unique and identifiable, facilitating accurate record retrieval, updating, and deletion. Foreign keys create connections between tables, maintaining referential integrity and ensuring that a value in one table corresponds to a value in another, thereby establishing reliable relationships throughout the database.

Primary Keys:

Customer Table: CustomerID - Uniquely identifies each customer.
Order Table: OrderID - Uniquely identifies each order.
Menu_Item Table: ItemID - Uniquely identifies each menu item.
Restaurant Table: RestaurantID - Uniquely identifies each restaurant.
Employee Table: EmployeeID - Uniquely identifies each employee.
Delivery Table: DeliveryID - Uniquely identifies each delivery.
Kitchen Table: KEmployeeID - Uniquely identifies each kitchen staff member.
Delivery Driver Table: DEmployeeID - Uniquely identifies each delivery driver.
Customer_Payment_Details Table: PaymentDetailID - Uniquely identifies each record of customer payment details.
Item Ingredients Table: IngredientID - Uniquely identifies each ingredient.
Inventory Table: InventoryID - Uniquely identifies each inventory record.
Promotion Table: PromotionID - Uniquely identifies each promotion.

Foreign Keys:

Order Table:
CustomerID: References Customer.CustomerID
RestaurantID: References Restaurant.RestaurantID

Employee Table:
RestaurantID: References Restaurant.RestaurantID

Delivery Table:
OrderID: References Order.OrderID
DeliveryPersonID: References Employee.EmployeeID

Order Item Table:
OrderID: References Order.OrderID
ItemID: References Menu_Item.ItemID

Kitchen Table:
KEmployeeID: References Employee.EmployeeID

Delivery Driver Table:
DEmployeeID: References Employee.EmployeeID

Customer_Payment_Details Table:
CustomerID: References Customer.CustomerID

Item Ingredients Table:
ItemID: References Menu_Item.ItemID

Inventory Table:
ItemID: References Menu Item.ItemID

Promotion Table:
ItemID: References Menu_Item.ItemID

Step 3: Creating Indexes:

Overview:

To expedite the retrieval of records from a database, indexes are essential tools because they offer quick paths to data. Databases can save a lot of time searching tables for pertinent rows by indexing frequently accessed or queried columns, like those used in WHERE clauses, JOIN conditions, and ORDER BY clauses.

Index Strategy:

Customer Table:

Indexes: Email, Phone
Purpose: Customers are often searched by contact information in customer service scenarios, so indexing these fields improves the speed of such queries.

Order Table:

Indexes: OrderDate, CustomerID
Purpose: Orders are commonly queried by date for reports and by customer ID for customer queries. Indexing these columns enhances performance for these operations.

Menu Item Table:

Indexes: Category
Purpose: Enhances performance when filtering menu items by categories in queries, making it quicker to retrieve specific types of menu items.

Employee Table:

Indexes: RestaurantID
Purpose: Useful for quickly finding all employees working at a specific restaurant, aiding in efficient staff management and query performance.

Delivery Table:

Indexes: OrderID, EstimatedTimeOfArrival
Purpose: Commonly used in tracking deliveries and calculating delivery performance. Indexing these columns helps speed up these processes.

Promotion Table:

Indexes: StartDate, EndDate
Purpose: Promotions are often queried based on their active dates. Indexing these fields can improve the performance of queries that filter promotions by date ranges.

Inventory Table:

Indexes: ItemID, SupplierName
Purpose: Frequently used for inventory checks and supplier queries. Indexing these columns can expedite these operations.

By optimizing the database for common use cases, these indexes guarantee effective data retrieval and improve system performance. The database can serve requests more quickly and enhance both backend operations and user experience by carefully placing indexes on fields that receive a lot of queries.

Step 4: Implementation in SQL:

Overview:

Creating tables, establishing relationships with keys, and optimizing access patterns with indexes are all part of implementing the database using SQL. To turn the theoretical design into a workable, functional database system, this step is essential.

SQL Commands:

Creating a Table (Example: Customer Table)

```
CREATE TABLE Customer (
  CustomerID INT AUTO_INCREMENT,
  Name VARCHAR(100),
  Address VARCHAR(255),
  Email VARCHAR(100),
  Phone VARCHAR(15),
  PRIMARY KEY (CustomerID)
);
```

Adding a Primary Key (In the table creation step above, the primary key is included.)

Establishing a Foreign Key Relationship (Example: Linking Order to Customer)

```
ALTER TABLE Order
ADD CONSTRAINT FK_Order_Customer
FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID);
```

Creating an Index (Example: Index on Email in Customer Table)

```
CREATE INDEX idx_Customer_Email
ON Customer (Email);
```

Step 5: Testing and Validation:

Overview:

Testing the database involves verifying that it operates accurately and efficiently in addition to adhering to the design. This phase guarantees optimal database performance under operational circumstances.

Testing Procedures:

Sample Data Insertion: Populating each table with sample data to mimic real operational data. For example, inserting varied customer profiles into the Customer table.

Testing Foreign Key Constraints: Ensuring that all foreign key relationships enforce data integrity. Attempt to insert an order with a non-existent customer ID to test if the database correctly blocks the insertion.

Index Performance Tests: Evaluate the performance of created indexes by executing queries that should benefit from these indexes, such as searching for customers by email.

Adjustments and Optimization:

Schema Adjustments: Based on testing outcomes, adjustments may be necessary. For example, increasing the size of the varchar fields if initial limits were too restrictive.

Performance Optimization: If certain queries are underperforming, additional indexes can be added by us or modify the existing ones to better support query patterns.

Conclusion:

The development of an app-based restaurant ordering database system is a multifaceted project that requires meticulous planning, design, and implementation. This project has provided a comprehensive approach to creating a robust database system to manage the various aspects of an online food ordering and delivery process. From the initial identification of business requirements to the final implementation and testing stages, each step has been designed to ensure the database meets the operational needs and enhances the overall efficiency and customer satisfaction.

Summary:

1. **Business Operation Task:**
    - We clearly defined the business operation tasks, identifying key processes such as customer interaction, order processing, and delivery or pickup.
    - Detailed the essential requirements for the database, ensuring it can handle customer management, menu management, order processing, restaurant location management, delivery and pickup tracking, employee management, inventory management, and promotions.

2. **Entity-Relationship Model (ER Model):**
    - Developed an ER model that accurately represents the entities involved in the online ordering system and their relationships.
    - Ensured the model includes four to six entities, focusing on critical areas like customers, orders, menu items, restaurants, employees, and deliveries.

3. **Relational Schema:**
    - Translated the ER model into a relational schema, ensuring it adheres to normalization principles up to the third normal form (3NF) to maintain data integrity and reduce redundancy.
    - Defined primary and foreign keys to establish clear relationships between tables.

4. **Database Implementation:**
    - Created detailed specifications for each table, including fields, data types, and constraints.
    - Defined and implemented indexes to optimize data retrieval, ensuring efficient query performance.
    - Provided SQL commands for creating tables, establishing key relationships, and indexing critical fields.

5. **Testing and Validation:**
    - Conducted thorough testing by inserting sample data, verifying foreign key constraints, and assessing index performance.
    - Made necessary adjustments based on testing outcomes to optimize database performance and ensure data integrity.

Benefits of the Database System:

- **Enhanced Customer Experience:** By storing comprehensive customer data, the system supports personalized interactions and efficient order management.
- **Efficient Order Processing:** Real-time order tracking and status updates ensure smooth coordination between the kitchen, delivery, and customer, leading to higher customer satisfaction.
- **Inventory and Menu Management:** The system provides up-to-date information on menu items and inventory levels, supporting effective stock management and menu customization.
- **Delivery Optimization:** Detailed tracking of deliveries and drivers ensures timely and accurate order fulfillment.
- **Employee Management:** Centralized employee data helps streamline scheduling and role assignment, ensuring adequate staffing.
- **Promotion Management:** The system supports targeted marketing efforts by managing promotions and analyzing their effectiveness.

Future Considerations:

- As the system is deployed and used, continuous monitoring and optimization will be necessary to address any emerging performance issues or evolving business requirements.
- Regular updates to the schema and indexing strategies may be needed to adapt to changes in order patterns, customer preferences, and technological advancements.
- Additional features, such as advanced analytics and integration with third-party delivery services, could further enhance the system's capabilities and provide valuable insights for business growth.

Data-Driven Decisions: SQL Queries for Enhancing Restaurant Business

Business Question 1: What are the most popular menu items?

Understanding which items are most frequently ordered can help in inventory planning and marketing strategies.

SQL Statement:

```
SELECT Name, OrderCount
FROM (SELECT MI.Name, COUNT(OI.ItemID) AS OrderCount,
ROW_NUMBER() OVER (ORDER BY COUNT(OI.ItemID) DESC) AS rn
FROM Menu_Item MI
JOIN Order_Item OI ON MI.ItemID = OI.ItemID
GROUP BY MI.Name
)
```

Business Question 2: What is the average order value?
Knowing the average order value can help in understanding customer spending habits and evaluating pricing strategies.
Benchmarking: Compare your AOV with industry standards or competitors to evaluate if your pricing strategy is effective.
Adjustment: If the AOV is lower than desired, consider adjusting pricing, offering bundled deals, or introducing premium items to encourage higher spending.

SQL Statement:

```
SELECT AVG(TotalAmount) AS AverageOrderValue
FROM "Order";
```

Business Question 3: Which customers are our most frequent diners?
Identifying customers who place orders frequently can help tailor customer loyalty programs and personalize marketing efforts. This query retrieves the names of customers and the count of orders they have placed. Sorting the results in descending order of Order Count highlights those who order most frequently.

SQL Statement:

```
SELECT C.CUSTOMER_NAME, COUNT(O.ORDERID) AS OrderCount
FROM CUSTOMER C
```

```
JOIN "Order" O ON C.CUSTOMER_ID = O.CUSTOMERID
GROUP BY C.CUSTOMER_NAME
ORDER BY OrderCount DESC;
```

Business Question 4: Which restaurant locations have the highest total sales? This helps in understanding the performance of each restaurant location, which is crucial for operational and financial planning.

SQL Statement:

```
SELECT R.RESTAURANT_NAME, SUM(O.TOTALAMOUNT) AS
TotalSales
FROM RESTAURANT R
JOIN "Order" O ON R.RESTAURANT_ID = O.RESTAURANTID
GROUP BY R.RESTAURANT_NAME
ORDER BY TotalSales DESC;
```

Business Question 5: Which employees have handled the most orders? Identifying top-performing employees can help in performance evaluations and reward programs.

SQL Statement:

```
SELECT E.NAME AS EmployeeName, COUNT(D.ORDERID) AS
OrdersHandled
FROM EMPLOYEE E
JOIN DELIVERY D ON E.EMPLOYEEID = D.DELIVERYPERSONID
GROUP BY E.NAME
ORDER BY OrdersHandled DESC;
```

Business Question 6:
This query calculates the average amount each customer spends per order, which helps in understanding customer value.

SQL Statement:

```
SELECT C.CUSTOMER_NAME, AVG(O.TOTALAMOUNT) AS
AverageSpend
FROM CUSTOMER C
JOIN "Order" O ON C.CUSTOMER_ID = O.CUSTOMERID
GROUP BY C.CUSTOMER_NAME
ORDER BY AverageSpend DESC;
```

Business Question 7: What ingredients are used for each menu item and in what quantities?
This query helps the business by optimizing inventory management and cost control, ensuring ingredient availability, and minimizing waste. It also maintains consistency and quality in menu item preparation, enhancing customer satisfaction and operational efficiency.

SQL Statement:

```
SELECT MI.NAME AS MenuItem, II.NAME AS Ingredient, II.QUANTITY
FROM MENU_ITEM MI
JOIN ITEM_INGREDIENTS II ON MI.ITEMID = II.ITEMID
ORDER BY MI.NAME, II.NAME;
```

Business Question 8: Which promotions are most effective in terms of revenue generated?
This query helps the business by identifying the promotions that generate the highest revenue, enabling better marketing and promotional strategies. It allows the business to focus on the most successful promotions, optimizing future campaigns to maximize sales and customer engagement.

SQL Statement:

```
SELECT DBMS_LOB.SUBSTR(P.DESCRIPTION, 100, 1) AS
PromotionDescription,
SUM(OI.QUANTITY * MI.PRICE * (1 - P.DISCOUNTPERCENTAGE /
100)) AS RevenueGenerated
FROM PROMOTION P
JOIN ORDER_ITEM OI ON P.ITEMID = OI.ITEMID
JOIN MENU_ITEM MI ON OI.ITEMID = MI.ITEMID
GROUP BY DBMS_LOB.SUBSTR(P.DESCRIPTION, 100, 1)
ORDER BY RevenueGenerated DESC;
```

Summary:

These business questions cover various aspects of the restaurant's operations, including sales performance, promotion effectiveness, employee productivity, and peak operational times. They provide valuable insights that can help in strategic decision-making and operational improvements. The SQL statements are designed to retrieve and analyse the data required to answer these questions effectively.