ASSIGNMENT/ASSESSMENT ITEM COVER SHEET

Student Name: MOHAMMED (FIRST NAME) MONTASIR RAHMAN (FAMILY / LAST NAME)

Student Number: 3 4 2 1 8 4 5 Email: Mohammed.MontasirRahman@uon.edu.au

Course Code	Course Title
C O M P 1 1 4 0	Database and Information Management
(Example) A B C D 1 2 3 4	*(Example)* Intro to University

Campus of Study: Callaghan (eg Callaghan, Ourimbah, Port Macquarie)

Assessment Item Title: Assignment 3 Due Date/Time: 25/10/2023

Tutorial Group (If applicable): EF108 3PM Word Count (If applicable):

Lecturer/Tutor Name: Suhuai Luo

Extension Granted: (•) Yes ( ) No Granted Until: 26/10/23

Please attach a copy of your extension approval

NB: STUDENTS MAY EXPECT THAT THIS ASSIGNMENT WILL BE RETURNED WITHIN 3 WEEKS OF THE DUE DATE OF SUBMISSION

Please tick box if applicable

☒ *Students within the Faculty of Business and Law, Faculty of Science and Information Technology, Faculty of Engineering and Built Environment and the School of Nursing and Midwifery:*
I verify that I have completed the online Academic Integrity Module and adhered to its principles

☒ *Students within the School of Education:*
"I understand that a minimum standard of correct referencing and academic literacy is required to pass all written assignments in the School of Education; and I have read and understood the School of Education *Course Outline Policy Supplement*, which includes important information related to assessment policies and procedures.

I declare that this assessment item is my own work unless otherwise acknowledged and is in accordance with the University's academic integrity policy available from the Policy Library on the web at http://www.newcastle.edu.au/policylibrary/000608.html
I certify that this assessment item has not been submitted previously for academic credit in this or any other course. I certify that I have not given a copy or have shown a copy of this assessment item to another student enrolled in the course.

I acknowledge that the assessor of this assignment may, for the purpose of assessing this assignment:
- Reproduce this assessment item and provide a copy to another member of the Faculty; and/or
- Communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the item on its database for the purpose of future plagiarism checking).
- Submit the assessment item to other forms of plagiarism checking.

DATE STAMP HERE

I certify that any electronic version of this assessment item that I have submitted or will submit is identical to this paper version.

Turnitin ID: (if applicable)

Insert this way

Signature:  Date: 26/09/2023

To copy and paste the completed form into another document use the 'snapshot' tool.

Print Form

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

COMP1140 S2 2023

Assignment 3

Project: Database design of Numberone Pizza

Final Report

Table of Contents

Preface

The database requirements for Number One Pizza involve gathering and storing diverse data through various interactions and events, including customer, employee, and supplier details. It encompasses order status and specifics, menu management with ingredient details, and overseeing company restocking orders. The database also tracks employee management, payments, and shifts. This comprehensive data forms the core of the system, enhancing decision-making speed and overall store efficiency.

Clear business regulations and methods for handling transactions are crucial for effective database management. When it comes to Order Processing, there are specific rules and procedures governing the recording and modification of order details, customer information, payment methods, and order statuses. In a similar manner, menu Items, Ingredients, Ingredient Orders, and Supplier relationships require well-defined rules and procedures to manage inventory levels, streamline reordering, and maintain connections with suppliers. For Employees, it's essential to have rules in place for managing shifts, compensation, and work hours to ensure all employees are productive, well-informed about their roles, and receive accurate pay.

The complexity of the database is showcased through the EER diagram, offering a visual representation of the intricate connections among data entities and their attributes, as well as the data flow, which plays a pivotal role in store management and operations.

This conceptual model has been developed through a thorough analysis of requirements. Subsequently, a relational model has been constructed and presented in the form of a DBML (Database Markup Language) representation. This relational model has undergone the process of normalization to achieve Boyce-Codd Normal Form (BCNF).

This report also delves into the process of creating the logical database design. It encompasses the steps of normalization and mapping, providing a comprehensive understanding of the database structuring process. Additionally, the report includes an EER model translated into the relational model, and it presents the various steps involved in the normalization process.

Furthermore, the report contains an SQL script that can be used to create a database for "NumberOne Pizza." Alongside this, it provides SQL statements essential for fulfilling transaction requirements.

Part 1: Reflection on Assignment 2

This section presents necessary discussion to point out the differences between my submitted report for assignment 2 and the current report.

In the data dictionary for Assignment 1, several attributes were initially missing. I have since rectified this issue by including all the previously omitted attributes in this subsequent part of the assignment.

Furthermore, during the relational mapping phase, I realized that while the primary keys of the tables were accurate, I had made errors in incorporating the foreign keys for some of the tables. In this assignment, I have concentrated on rectifying these discrepancies, ensuring that all necessary foreign keys have been correctly added.

It's worth noting that I neglected to mention in the previous assignment that all tables are in Boyce-Codd Normal Form (BCNF) except for two of them. In this assignment, I have provided an explanation for the non-BCNF status of these two tables, thereby addressing this omission.

Lastly, in the example of normalization up to BCNF, I had previously only presented one illustrative case. To improve the comprehensiveness of the assignment, I have included an additional example of normalization, thereby enriching the content.

Throughout the following discussions, I have diligently rectified the previous assignment's shortcomings and continued with accurate and essential explanations.

Part 2: Requirements

Data Requirements

For Order Processing:

In the order processing system for "Numberone Pizza," a diverse range of data must be gathered to guarantee the smooth and precise handling of customer orders. Customers can order through Phone Order) and walk in to the restaurant which is (Walkin Order). The call received time(receivedTime) and termination time(terminatedTime) for phone orders are recorded. Also, for walk in order, the walk in time (walkinTime) and food pick up time (pickupTime) is recorded in the database. The pick up time(pickupTime) is also recorded for the phone orders as well, as there is an option to pick up along with delivery. For delivery orders, the total delivery time(deliveryTime) , address of food delivery(deliveryAddress) and the name of delivery driver (driverName) is recorded.

When a customers orders the following details are recorded (orderDate), (customerName), (customerPhone), (customerAddress), (staffID), (itemOrdered), (quantiesOfOrderedItem), (priceofItems), (totalAmount), (paymentMethod), (paymentApprovalNo), (orderStatus), (orderType), and (orderDescription).

Menu Items, Ingredients, Ingredient Order and Suppliers:

The system should additionally gather information about menu items and ingredients. Each menu item should possess a distinct item code (itemCode), a name (itemName), a size (itemSize), and a price (itemPrice). For each menu item, the system should keep a record of the required ingredients (ingredients) and their corresponding quantities (ingredientAmount). Similarly, just as with menu items, data about ingredients should encompass an exclusive code (ingredientCode), a name (ingredientName), a type (ingredientType), a stock unit (stockUnit), the supplier responsible for the ingredient (ingredientSupplier), and a description (ingredientDescription) of the ingredient.

A weekly stock take is conducted where the quantity of an Ingredient is measured and recorded (actualStockAmount) based on its weight. This recorded amount is then compared against the suggested stock level (suggestedStockAmount). If the ingredient's weight is lower than the suggested amount, an order for restocking must be initiated. The date of the last stocktake (StockTakeDate) is documented.

In the database, a list of suppliers providing the ingredient is stored SupplierList, including their phone number (supplierPhone),suppliers unique id(supplierId), email (supplierEmail), address (supplierAddress), the supplied ingredient (ingredientSupplier), and name (supplierName). When ordering a restock, the order date (sOrderDate), receipt date (sOrderReceived), and order status (sOrderStatus) are recorded. The database also holds data about the price of all ingredients (ingredientPrice), total price (sOrderTotalPrice),quantity of ingredients (ingredientQuantities), description of stock (stockDescription) and order number (sOrderNumber).

Employees:

In the 'Number One Pizza' database, every employee is identified uniquely (staffId). Additional employee data includes first name(firstName),last name(lastName), address (eAddress),phone number (eNumber), employee status and description (eStatus),(eDescription).Tax info requires the employee's tax file number (taxNumber).

For payments, bank details are stored, like bank name (bankName), bank code (bankCode), and account number (accountNumber). Payment records cover gross payment (grossPay), tax withheld (taxWithHeld), payment date (payDate), and payment period (payPeriodStart, payPeriodEnd.

Roles are instore staff(InstoreStaff) or delivery drivers (DeliveryStaff). Driver's licenses (driverLicense) are noted for drivers. Shifts include start and end times (startDate,startTime,endDate,endTime), shift details (Shift), shift hours (workHours),hourly pay for in-store staff, staffs payment (staffPayment) and delivery driver (paymentRate) is stored. For delivery vehicles, number of deliveries by a driver in a shift (noOfDelivery), registration numbers (regoNumber) are stored.

Transaction Requirements

Data Manipulation

- Insert, Update and Delete existing order
- Insert, Update and Delete customer information
- Update orderStatus
- Insert, delete, and update supplier details from the supplier list.
- Comparing the actual stock amount of ingredients with the suggested stock amount during stock take.
- Retrieving details about ingredients, such as ingredient code, name, type, stock unit, supplier, and description.
- Getting data about suppliers, including their contact information and supplied ingredients.
- Adding new records for menu items, ingredients, and suppliers to the respective tables.
- Calculating the total price of ingredients in an order based on their prices and quantities.
- Adding new employee records with their details, roles, and payment information.
- Updating employee status and description.
- Retrieve details of a particular shift such as shift time
- Insert, Update and Delete a particular staffs pay

Queries

- Lookup a customer using an existing customer's phone number.
- Choose between delivery or walk-in for the customer's order.
- Indicate whether the customer is making payment with credit, debit, or cash.
- Predict the time required for an order to reach the customer's location.
- Document the time taken to prepare and deliver the order to the customer.
- Weigh current stock and compare to recommended stock weight.

- Contact supplier and order new stock.
- Calculate how many deliveries a delivery driver has made per shift.
- See whether an employee is working as a in-store staff or as a delivery driver.

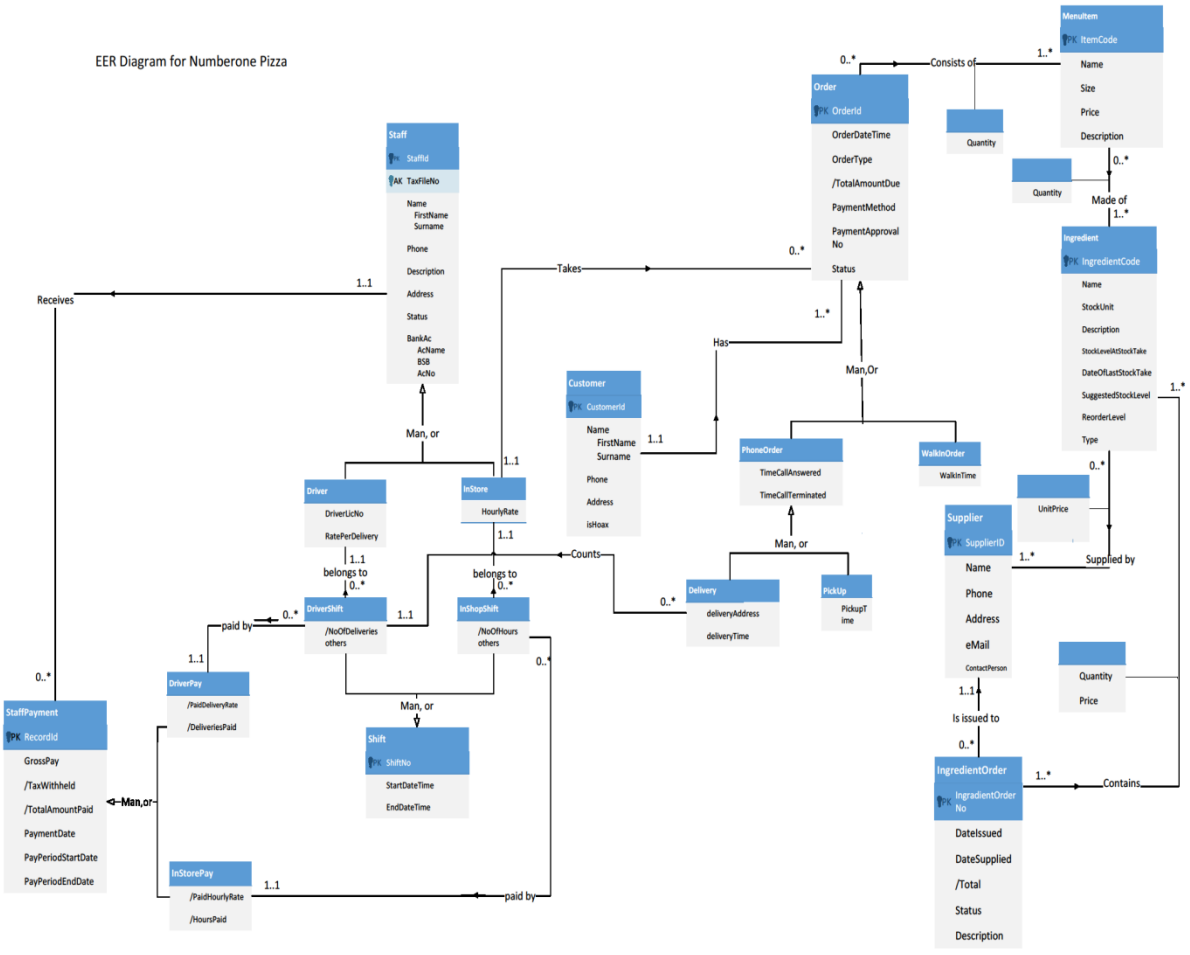
Business Rules

1. Orders can be placed either in person or over the phone.
2. Customers have the option for either picking up their orders or having them delivered.
3. The system records the timing of each call.
4. Suppliers provide one or more types of ingredients.
5. Employees are categorized as either in-store staff or drivers.
6. Ensure that employees are punctual and adhere to their assigned shifts.
7. Accurately record orders, including any customizations and special requests.
8. Maintain a clear and regularly updated menu with accurate pricing information.
9. The system should offer online order placement for customers.
10. Customers should have the ability to peruse the menu, personalize their orders, and securely complete payments.
11. The system should generate various reports, including summaries of sales, popular pizza items, and customer order histories.
12. Managers should be able to analyze sales patterns to make well-informed decisions.
13. The system should grant different levels of access to employees, managers, and administrators.
14. Each user should possess a unique username and password for authentication.

Part 3: EER Model and Data dictionary

EER Model

c3421845



Data Dictionary

Entity:

Entity	Description	Aliases	Occurrences
Customer	Who places the order and to whom the finished order is served or delivered.	• Order • DeliveryStaff • CompleteOrder	• When a customer creates an order
Order	Describes the purchases made by customers.	• Customers • Menu	• When a customer places an order • When they present their order to the staff • When they choose their order from the menu.
Menu	The selection list from which the customer chooses the products for their purchase.	• Order • Ingredients	• When the customer selects the things they want to order. • When the components are turned into dishes on the menu.
EmployeeShift	Entityt that generates and fulfils consumer orders	• Order • Shift • In-store staff • Delivery staff • payment	• When the employee receives the order. • When a worker does a shift
InstoreStaff	Orders are made by store employees who are paid.	• Employee • Shift	• When staff works in store
DeliveryStaff	Employees,drive to deliver orders to customers	• Employee • DeliveryVehicle	• Where DeliveryStaff are an employee category.

Ingredient	Ingredients required to make the dishes ordered from the menu	• Menu • Stock	• Where ingredients are used to create the menu items
SupplierList	The list of suppliers information stored	• Stock	• When ingredient needs to be ordered • Check what has been received
PaymentRecord	Payment information kept	• Employee • Payment	• When an employee is paid
Employee	Employee who works in store and as a driver	• DeliveryStaff • Instore staff	• To see employee details
Orderitem	Customer order quantity of food	• Customer	• When a customer orders pizza
InstoreStaff	Staff's workhours and payment rate	• Customer	• To see instore staff details
Phone Order	The phone order made by a customer	• Order	• To see details of phone order
Walkin Order	When customer entering the shop to place an order	• Order	• To see details of walkin order
DriverShift	Drivers work hours and no. of deliveries	• Delivery Staff • Delivery Vehicle	• Drivers shift start and end time
InShopShift	Shift of employee who works in store	• Instore Staff	• See instore employee shift start and end time
DriverPay	Drivers pay based on number of deliveries	• Vehicle • Driver	• Check payrate for number of deliveries
InStorePay	Pay of the employees in the store	• Instore Staff • Wakin order	• Check payrate for instore staff
Delivery	Number of deliveries has been made	• Vehicle • Delivery Staff • Delivery	• See Information about a delivery
QOrderMenuItem	Quantities of order from the menu	• Menu Item • Order	• Quantity of Items needed for an order

QMenuItemIngredient	Quantities of ingredient for a menu item	• Menu Item • Ingredient	• Quantites of ingredient needed for a menu item
QIngredientIngOrder	Quantities of ingredient needed for an order	• Ingredient • Order	• When quantites of ingredient needed for an order to be checked
QSupplierIngredient	Quantities of Ingredient supplied by supplier	• Supplier details • Ingredient	• Find which supplier delivers an ingredient
Pickup	The name and time of customer when picked up	• Pickup • Customer • Order	• See number of pickups in a shift

Relationships:

Entity Name	Multiplicity	Relationship	Multiplicity	Entity Name
Customer	1..1	Has	1..*	Order
Order	0..*	Consists of	1..*	MenuItem
	(Man, Or)	Generalization	(Man, Or)	WalkInOrder
	(Man, Or)	Generalization	(Man, Or)	PhoneOrder
MenuItem	0..*	Made of	1..*	Ingredient
IngredientOrder	1..*	Contains	1..*	Ingredient
	0..*	Is Issued to	1..1	Supplier
Ingredient	0..*	Supplied by	1..*	Supplier
PhoneOrder	(Man, Or)	Generalization	(Man, Or)	Delivery
	(Man, Or)	Generalization	(Man, Or)	PickUp
Delivery	0..*	Counts	1..1	DriverShift
InStore	1..1	Takes	0..*	Order
Staff	1..1	Receives	0..*	StaffPayment
	(Man, Or)	Generalization	(Man, Or)	Driver
	(Man, Or)	Generalization	(Man, Or)	InStore
InShopShift	0..*	belongs to	1..1	InStore
	0..*	paid by	1..1	InStorePay
DriverShift	0..*	belongs to	1..1	Driver
	0..*	paid by	1..1	DriverPay
Shift	(Man, Or)	Generalization	(Man, Or)	DriverShift
	(Man, Or)	Generalization	(Man, Or)	InShopShift
StaffPayment	(Man, Or)	Generalization	(Man, Or)	DriverPay
	(Man, Or)	Generalization	(Man, Or)	InStorePay

Attributes:

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Order	OrderNo	Unique order identifier	char(10)	N	N	N	
	OrderDateTime	The date & time the order is made	datetime	Y	N	N	
	OrderType	Walkin or Phone order	varchar(10)	Y	N	N	
	TotalAmountDue	Total money for the order	float	Y	N	N	
	PaymentMethod	Total 3 kinds of payment	varchar(10)	Y	N	N	
	PaymentApprovalNo	Payment Approval No	varchar(10)	Y	N	N	
	Status	Current state of the order, including executed or not	varchar(10)	Y	N	N	
MenuItem	ItemCode	Unique code or identifier assigned to the menu item	char(10)	N	N	N	
	Name	Name of the menu item	varchar(10)	Y	N	N	
	Size	Size or portion options available for the menu item	varchar(10)	Y	N	N	
	Price	Cost or price associated with ordering the menu item	float	Y	N	N	

	Description	Brief text that provides additional information or details about the menu item	varchar(10)	Y	N	N	
Ingredient	IngredientCode	Unique code assigned to the ingredient for internal tracking	char(10)	N	N	N	
	Name	Name of the ingredient	varchar(10)	Y	N	N	
	StockUnit	Measurement used to quantify and manage the ingredient's stock	char(10)	Y	N	N	
	Description	Description that provides additional information about the ingredient	varchar(10)	Y	N	N	
	StockLevelAtStockTake	Quantity of the ingredient on hand as determined during the most recent stocktake	char(10)	Y	N	N	
	DateOfLastStockTake	Date when the last stocktake was conducted for the ingredient	datetime	Y	N	N	
	SuggestedStockLevel	Recommended quantity of the ingredient that should be maintained in stock	char(10)	Y	N	N	
	ReorderLevel	Stock level at which it is advisable to initiate a reorder	char(10)	Y	N	N	
	Type	Classifies the ingredient into relevant types or categories	varchar(10)	Y	N	N	

Supplier	SupplierID	Unique identifier assigned to the supplier for internal tracking	char(10)	N	N	N	
	Name	Name of the supplier	varchar(10)	Y	N	N	
	Phone	Number at which the supplier can be reached for inquiries	char(10)	Y	N	N	
	Address	Physical location or mailing address of the supplier	varchar(10)	Y	N	N	
	eMail	Email address associated with the supplier	varchar(10)	Y	N	N	
	ContactPerson	Name of an individual within the supplier's organization who serves as the primary point of contact for inquiries	varchar(10)	Y	N	N	
Ingredient Order	IngredientOrderNo	Unique identifier assigned to the ingredient order for tracking	char(10)	N	N	N	
	DateIssued	Date when the ingredient order was created	datetime	Y	N	N	
	DateSupplied	Date when the ingredients were actually delivered	datetime	Y	N	N	
	Total	Total cost associated with the ingredient order	float	Y	N	N	
	Status	Status indicator that reflects the current state of the ingredient order	varchar(10)	Y	N	N	
	Description	Notes related to the ingredient order	varchar(10)	Y	N	N	

PhoneOrder	TimeCallAnswered	Recorded to track the start of the order process.	varchar(10)	Y	N	N	
	TimeCallTerminated	Recorded to track the end of the order process	varchar(10)	Y	N	N	
WalkInOrder	WalkInTime	Records the moment the order process began for walk-in customers	varchar(10)	Y	N	N	
Delivery	deliveryAddress	Address to which the delivery should be made	varchar(10)	Y	N	N	
	deliveryTime	Expected time for the delivery to take place	datetime	Y	N	N	
PickUp	PickupTime	Time when the customer plans to arrive and pick up	datetime	Y	N	N	
Customer	CustomerId	Unique Id of Customer	char(10)	N	N	N	
	FirstName	First Name of customer	varchar(10)	Y	N	N	
	Surname	Last Name of customer	varchar(10)	Y	N	N	
	Phone	Phone Number of customer	char(10)	Y	N	N	
	Address	Address of customer	varchar(10)	Y	N	N	
	StaffId	Staff number assigned to each staff member	char(10)	N	N	N	
	TaxFileNo	Staff member's tax file number	char(10)	N	N	N	
	FirstName	Staff member's first name	varchar(10)	Y	N	N	

Staff	Surname	Staff member's last name	varchar(10)	Y	N	N	
	Phone	Number at which the staff member can be contacted	char(10)	Y	N	N	
	Description	Notes related to the staff member	varchar(10)	Y	N	N	
	Address	Physical address of the staff member	varchar(10)	Y	N	N	
	Status	Staff member's current employment status	varchar(10)	Y	N	N	
	AcName	Name associated with the staff member's bank account	varchar(10)	Y	N	N	
	BSB	Specific bank and branch where the staff member's bank account is held	char(4)	Y	N	N	
	AcNo	Account number associated with the staff member's account	char(15)	Y	N	N	
InStore	HourlyRate	Hourly wage assigned to an in-store employee	float	Y	N	N	
Driver	DriverLicNo	Driver's license number which is unique	char(13)	N	N	N	
	RatePerDelivery	Amount paid to the driver for each delivery they make	float	Y	N	N	
InShopShift	NoOfHours	Number of hours worked by an in-shop employee	float	Y	N	N	

DriverShift	NoOfDeliveries	Number of deliveries completed by a driver during a specific shift	varchar(10)	Y	N	N	
Shift	ShiftNo	Number assigned to the shift		N	N	N	
	StartDateTime	Time when the work assignment starts	datetime	Y	N	N	
	EndDateTime	Time when the work assignment concludes	datetime	Y	N	N	
DriverPay	PaidDeliveryRate	Amount paid to the driver for each delivery	float	Y	N	N	
	DeliveriesPaid	Total number of deliveries for which the driver has been paid	float	Y	N	N	
InStorePay	PaidHourlyRate	Rate of pay that an in-store employee receives for their work	float	Y	N	N	
	HoursPaid	Number of hours for which the in-store employee has been paid	float	Y	N	N	
StaffPayment	RecordId	Unique number assigned to the staff payment record	char(10)	N	N	N	
	GrossPay	Total amount of pay before any deductions	float	Y	N	N	
	TaxWithHeld	Amount of tax or taxes withheld from the staff	float	Y	N	N	
	TotalAmountPaid	Final amount paid to the staff member after deductions	float	Y	N	N	
	PaymentDate	Date of payment to the staff member	datetime	Y	N	N	

	PayPeriodStartDate	Start date of the pay period	datetime	Y	N	N	
	PayPeriodEndDate	End date of the pay period	datetime	Y	N	N	
	Quantity	Amount of item being purchased or sold	varchar(10)	Y	N	N	
	UnitPrice	Price of a single unit	float	Y	N	N	
	Price	Price of a specified quantity of item	float	Y	N	N	

Part 4: Mapping the EER to Relational Model

Using the mapping rules, got the following relations for all entities in EER.

=Customer (CustomerId, FirstName, Surname, Phone, Address)

Primary key CustomerId

=Staff (StaffId, TaxFileNo, FirstName, Surname, Phone, Description, Address, Status, AcName, BSB, AcNo)

Primary Key StaffId

=InStore (HourlyRate, ShiftNo)

Primary Key StaffId

Foreign Key StaffId references Staff (StaffId) ON UPDATE CASCADE, ON DELETE CASCADE

=Driver (DriverLicNo, RatePerDelivery, ShiftNo)

Primary Key StaffId

Foreign Key StaffId references Staff (StaffId) ON UPDATE CASCADE, ON DELETE CASCADE

=Orders (OrderId, OrderDateTime, OrderType, TotalAmountDue, PaymentMethod, PaymentApprovalNo, Status, CustomerId, StaffId)

Primary Key OrderId

Foreign Key CustomerId references Customer(CustomerId) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key StaffId references Instore (StaffId) ON UPDATE CASCADE, ON DELETE CASCADE

=WalkInOrder (OrderId, WalkInTime)

Primary Key OrderId

Foreign Key OrdeId references Orders (OrderId) ON UPDATE CASCADE, ON DELETE CASCADE

=PhoneOrder (OrderId, timeCallAnswered,TimeCallTerminated)

Primary Key OrderId

Foreign Key OrderId references Orders (OrderId) ON UPDATE CASCADE, ON DELETE CASCADE

=DeliveryOrder (OrderId, deliveryAddress, deliveryTime, ShiftNo)

Primary Key OrderId

Foreign Key OrderId references PhoneOrder (OrderId) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key ShiftNo references DriverShift (ShiftNo) ON UPDATE CASCADE, ON DELETE CASCADE

=PickupOrder (OrderId, deliveryAddress,deliveryTime)

Primary Key OrderId

Foreign Key OrderId references PhoneOrder (OrderId) ON UPDATE CASCADE, ON DELETE CASCADE

=MenuItem (ItemCode, Name, Size, Price, Description)

Primary Key ItemCode

=Ingredient (IngredientCode, Name, StockUnit, Description, StockLevelAtStockTake,

DateOfLastStockTake, SuggestedStockLevel, ReorderLevel, Type)

Primary Key IngredientCode

=QOrderMenuItem (ItemCode, OrderDate, QuantityOrdered, OrderId)

Primary Key ItemCode

Foreign Key ItemCode references MenuItem (ItemCode) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key OrderId references Order (OrderId) ON UPDATE CASCADE, ON DELETE CASCADE

=QMenuItemIngredient (ItemCode, IngredientCode, QuantityUsed)

Primary Key ItemCode

Foreign Key ItemCode references MenuItem (ItemCode) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key IngredientCode references Ingredient (IngredientCode) ON UPDATE CASCADE, ON DELETE CASCADE

=StaffPayment (RecordId, GrossPay, TaxWithHeld, TotalAmountPaid, PaymentDate, PayPeriodStartDate, PayPeriodEndDate, StaffId)

Primary Key RecordId

Foreign Key StaffId references Staff (StaffId) ON UPDATE CASCADE, ON DELETE CASCADE

=InStorePay (PaidHourlyRate, HoursPaid, RecordId)

Primary Key RecordId

Foreign Key RecordId references StaffPayment (RecordId) ON UPDATE CASCADE, ON DELETE CASCADE

=DriverPay (PaidDeliveryRate, DeliveriesPaid, RecordId)

Primary Key RecordId

Foreign Key RecordId references StaffPayment (RecordId) ON UPDATE CASCADE, ON DELETE CASCADE

=Shift (ShiftNo, StartDateTime, HoursPaid)

Primary Key ShiftNo

=DriverShift (NoOfDeliveries, ShiftNo, RecordId, StaffId)

Primary Key ShiftNo

Foreign Key ShiftNo references Shift (ShiftNo) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key RecordId references DriverPay (RecordId) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key StaffId references Driver (StaffId) ON UPDATE CASCADE, ON DELETE CASCADE

=InShopShift (NoOfHours, ShiftNo, RecordId, StaffId)

Primary Key Shift No

Foreign Key ShiftNo references Shift (ShiftNo) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key RecordId references InStorePay (RecordId) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key StaffId references InStore (StaffId) ON UPDATE CASCADE, ON DELETE CASCADE

=Supplier (SupplierID, Name, Phone, Address, eMail, ContactPerson)

Primary Key SupplierID

=IngredientOrder (IngredientOrderNo, DateIssued, DateSupplied, Total, Status, Description, SupplierID)

Primary Key IngredientOrderNo

Foreign Key SupplierID references Supplier (SupplierID) ON UPDATE CASCADE, ON DELETE CASCADE

=QIngredientIngOrder (IngredientOrderNo, IngredientCode, QuantityOrdered)

Primary Key IngredientOrderNo

Foreign Key IngredientOrderNo references IngredientOrder (IngredientOrderNo) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key IngredientCode references Ingredient (IngredientCode) ON UPDATE CASCADE, ON DELETE CASCADE

=QIngredientSupplier (SupplierID, IngredientOrderNo, PricePerUnit)

Primary Key IngredientOrderNo

Foreign Key SupplierID references Supplier(SupplierID) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key IngredientOrderNo references IngredientOrder (IngredientOrderNo) ON UPDATE CASCADE, ON DELETE CASCADE

Part 5 : Normalizing the Scheme up to BCNF

According to the definitions of 1NF, 2NF, 3NF and BCNF, it is identified that relations Customer, Staff, Instore, Driver, WalkInOrder, PhoneOrder , DeliveryOrder, PickupOrder, MenuItem, QOrderMenuItem, QMenuItemIngredient, StaffPayment, InstorePay, Supplier,

QIngredientIngOrder, QIngredientSupplier are all in BCNF, since all the attributes are atomic, and there exists only one function dependency in each table, and the left side of the FD is a PK.

But the following relations are not in BCNF. They are normalized as below.

Orders (OrderId, OrderDateTime, TotalAmountDue, PaymentMethod, PaymentApprovalNo, Status, CustomerId, StaffId)

Primary Key OrderId

Foreign Key CustomerId references Customer(CustomerId) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key StaffId references Instore(StaffId) ON UPDATE CASCADE, ON DELETE CASCADE

FD1: OrderId -> OrderDateTime, TotalAmountDue, PaymentMethod, PaymentApprovalNo, Status

FD2: PaymentApprovalNo -> PaymentMethod, TotalAmountDue

Therefore, there exists transitive dependency, so the relation is in 2nd but not 3rd NF.

The normalization process:

OrderOnly (OrderNo, OrderDateTime, PaymentApprovalNo, Status, CustomerId, StaffId)

Primary Key OrderNo

Foreign Key CustomerId references Customer(CustomerId) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key StaffId references Instore(StaffId) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key PaymentApprovalNo references OrderPayRecord (PaymentApprovalNo) ON UPDATE CASCADE, ON DELETE CASCADE

OrderPayRecord (PaymentApprovalNo, PaymentMethod, TotalAmountDue)

Primary Key PaymentApprovalNo

Note: Since the Orders is split into 2 tables, the relationships of original Orders to all the other tables should be reconsidered now – use OrderOnly to connect to other tables. Then the changed tables are:

QOrderMenuItem (Quantity, OrderNo, ItemCode)

Primary Key (OrderNo, ItemCode, Quantity)

Foreign Key OrderNo references OrderOnly(OrderNo) ON UPDATE CASCADE, ON DELETE CASCADE

Foreign Key ItemCode references MenuItem (ItemCode) ON UPDATE CASCADE, ON DELETE CASCADE

Ingredient Table:

Original Attributes: IngredientCode (Primary Key), Name, StockUnit, Description, StockLevelAtStockTake, DateOfLastStockTake, SuggestedStockLevel, ReorderLevel, Type

Functional Dependencies (FDs):

FD1: IngredientCode -> Name, StockUnit, Description, StockLevelAtStockTake, DateOfLastStockTake, SuggestedStockLevel, ReorderLevel, Type

The Ingredient Table is already in 2nd and 3rd Normal Form (NF) since IngredientCode is the primary key, and there are no partial dependencies.

Supplier Table:

Original Attributes: SupplierID (Primary Key), Name, Phone, Address, Email, ContactPerson
Functional Dependencies (FDs):

FD1: SupplierID -> Name, Phone, Address, Email, ContactPerson

The Supplier Table is already in 2nd and 3rd Normal Form (NF) since SupplierID is the primary key, and there are no partial dependencies.

Staff Table:

Original Attributes: StaffId (Primary Key), FirstName, Surname, Phone, Description, Address, AccountName, BSB, AccNum

Functional Dependencies (FDs):

FD1: StaffId -> FirstName, Surname, Phone, Description, Address, AccountName, BSB, AccountNumber

The Staff Table is already in 2nd and 3rd Normal Form (NF) since StaffId is the primary key, and there are no partial dependencies.

Shift Table:

Original Attributes: ShiftNo (Primary Key), StartDateTime, EndDateTime, StaffId (Foreign Key)
Functional Dependencies (FDs):

FD1: ShiftNo -> StartDateTime, EndDateTime, StaffId

The Shift Table is already in 2nd and 3rd Normal Form (NF) since ShiftNo is the primary key, and there are no partial dependencies.

StaffPayment Table:

Original Attributes: RecordId (Primary Key), GrossPay, TaxWithheld, TotalAmountPaid, PaymentDate, PayPeriodStartDate, PayPeriodEndDate, StaffId (Foreign Key)

Functional Dependencies (FDs):

FD1: RecordId -> GrossPay, TaxWithheld, TotalAmountPaid, PaymentDate, PayPeriodStartDate, PayPeriodEndDate, StaffId The StaffPayment Table is already in 2nd and 3rd Normal Form (NF) since RecordId is the primary key, and there are no partial dependencies.

IngredientOrder Table:

Original Attributes: IngredientOrderNo (Primary Key), DateIssued, Total, Status, Description, SupplierId (Foreign Key) Functional Dependencies (FDs):

FD1: IngredientOrderNo -> DateIssued, Total, Status, Description, SupplierId

The IngredientOrder Table is already in 2nd and 3rd Normal Form (NF) since IngredientOrderNo is the primary key, and there are no partial dependencies.

SQL SCRIPTS

```
DROP TABLE QIngredientSupplier
DROP TABLE QIngredientIngOrder
DROP TABLE QMenuItemIngredient
DROP TABLE QOrderMenuItem
DROP TABLE IngredientOrder
DROP TABLE InShopShift
DROP TABLE InStorePay
DROP TABLE PickupOrder
DROP TABLE DeliveryOrder
DROP TABLE PhoneOrder
DROP TABLE WalkInOrder
DROP TABLE DriverShift
DROP TABLE DriverPay
DROP TABLE Orders
DROP TABLE InStore
DROP TABLE Driver
DROP TABLE Shift
DROP TABLE StaffPayment

DROP TABLE Ingredient
DROP TABLE Supplier
DROP TABLE MenuItem
DROP TABLE Staff
DROP TABLE Customer

-- Create a table to store customer information
CREATE TABLE Customer (
    CustomerId CHAR(3) PRIMARY KEY,   -- Unique identifier for each customer (Eg: xxx)
    FirstName VARCHAR(50) NOT NULL,   -- First name of the customer (must not be null)
```

```
    Surname VARCHAR(50) NOT NULL,      -- Last name of the customer (must not be null)
    Phone VARCHAR(15) NOT NULL,        -- Customer's contact phone number (must not be
null)
    Address VARCHAR(255) NOT NULL      -- Customer's physical address (must not be null)
);

-- Create a table to store staff information
CREATE TABLE Staff (
    StaffId CHAR(2) PRIMARY KEY,       -- Unique identifier for each staff member (Eg: xx)
    TaxFileNo VARCHAR(15) NOT NULL,          -- Tax file number of the staff member (must not
be null)
    FirstName VARCHAR(50) NOT NULL,          -- First name of the staff member (must not be
null)
    Surname VARCHAR(50),           -- Last name of the staff member
    Phone VARCHAR(15),             -- Staff member's contact phone number
    Description VARCHAR(255),      -- Staff be Manager, Crew Member, or Cashier
    Address VARCHAR(255),          -- Staff member's physical address
    Status VARCHAR(20),            -- Staff member's employment status (e.g., active, inactive)
    AcName VARCHAR(50),            -- Bank account name for payroll purposes
    BSB VARCHAR(10),               -- Bank State Branch code for direct deposits
    AcNo VARCHAR(15)               -- Bank account number for direct deposits
);

-- Create a table to store MenuItem information
CREATE TABLE MenuItem (
    ItemCode CHAR(5) PRIMARY KEY,   -- Unique identifier for the item
    Name VARCHAR(255) NOT NULL,      -- Name of the menu item (required)
    Size VARCHAR(50),                -- Size of the item (optional)
    Price DECIMAL(10, 2) NOT NULL,    -- Price of the item (required)
    Description TEXT                 -- Description of the item
);
```

```
--Supplier table for recording supplier information
CREATE TABLE Supplier (
    SupplierID CHAR(5) PRIMARY KEY, -- Unique identifier for the supplier
    Name VARCHAR(255) NOT NULL,      -- Name of the supplier (required)
    Phone VARCHAR(20),               -- Phone number of the supplier
    Address VARCHAR(255),            -- Address of the supplier
    eMail VARCHAR(255),              -- Email address of the supplier
    ContactPerson VARCHAR(100)       -- Name of the contact person at the supplier
);

--Ingredient table for recording ingredient information
CREATE TABLE Ingredient (
    IngredientCode CHAR(5) PRIMARY KEY,        -- Unique identifier for the ingredient
    Name VARCHAR(255) NOT NULL,                -- Name of the ingredient (required)
    StockUnit VARCHAR(50),                     -- Unit of measurement for stock
    Description TEXT,                          -- Description of the ingredient
    StockLevelAtStockTake DECIMAL(10, 2),      -- Stock level at the last stock take
    DateOfLastStockTake DATE,                  -- Date of the last stock take
    SuggestedStockLevel DECIMAL(10, 2),        -- Suggested stock level for the ingredient
    ReorderLevel DECIMAL(10, 2),               -- Reorder level for the ingredient
    Type VARCHAR(50)                           -- Type or category of the ingredient
);

-- InStore table
CREATE TABLE InStore (
    StaffId CHAR(2),
    HourlyRate DECIMAL(10, 2) DEFAULT 15.00,  --Default value of 15.00
    ShiftNo INT,
    PRIMARY KEY (StaffId),
```

```
    FOREIGN KEY (StaffId) REFERENCES Staff (StaffId) ON UPDATE CASCADE ON DELETE
CASCADE
);

-- Driver table
CREATE TABLE Driver (
    StaffId CHAR(2),
    DriverLicNo INT,
    RatePerDelivery DECIMAL(10, 2),
    ShiftNo INT,
        PRIMARY KEY ( StaffId ),
    FOREIGN KEY (StaffId) REFERENCES Staff (StaffId) ON UPDATE CASCADE ON DELETE
CASCADE
);

--Create the Shift table
CREATE TABLE Shift (
    ShiftNo INT PRIMARY KEY,       -- Unique identifier for each shift
    StartDateTime DATETIME,        -- Date and time when the shift starts
    HoursPaid DECIMAL(5, 2)        -- Number of hours paid for the shift
);

--Create the Orders table
CREATE TABLE Orders (
    OrderId CHAR(5) PRIMARY KEY,
    OrderDateTime DATETIME,
    OrderType VARCHAR(50),
    TotalAmountDue DECIMAL(10, 2) DEFAULT 0.00, -- Default value 0.00
    PaymentMethod VARCHAR(50),
    PaymentApprovalNo VARCHAR(50) DEFAULT NULL, -- Default value NULL
    Status VARCHAR(50),
```

```
    CustomerId CHAR(3),
    StaffId CHAR(2),
    FOREIGN KEY (CustomerId) REFERENCES Customer(CustomerId) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (StaffId) REFERENCES Instore(StaffId) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Create the WalkInOrder table
CREATE TABLE WalkInOrder (
    OrderId CHAR(5) PRIMARY KEY,
    WalkInTime DATETIME,
    FOREIGN KEY (OrderId) REFERENCES Orders(OrderId) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Create the PhoneOrder table
CREATE TABLE PhoneOrder (
    OrderId CHAR(5) PRIMARY KEY,
    TimeCallAnswered DATETIME,
    TimeCallTerminated DATETIME,
    FOREIGN KEY (OrderId) REFERENCES Orders(OrderId) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Create the StaffPayment table
CREATE TABLE StaffPayment (
    RecordId INT PRIMARY KEY,
    GrossPay DECIMAL(10, 2) DEFAULT 0.00, -- Default value 0.00
    TaxWithHeld DECIMAL(10, 2) DEFAULT 0.00, -- Default value 0.00
    TotalAmountPaid DECIMAL(10, 2),
    PaymentDate DATE,
```

```
    PayPeriodStartDate DATE,
    PayPeriodEndDate DATE,
    StaffId CHAR(2),
    FOREIGN KEY (StaffId) REFERENCES Staff(StaffId) ON UPDATE CASCADE ON DELETE
CASCADE
);

--DriverPay table for staff payments related to delivery work
CREATE TABLE DriverPay (
    PaidDeliveryRate DECIMAL(10, 2), -- The rate paid to the staff for deliveries
    DeliveriesPaid INT,              -- The number of deliveries for which the staff is paid
    RecordId INT PRIMARY KEY,        -- Unique identifier for the payment
    FOREIGN KEY (RecordId) REFERENCES StaffPayment (RecordId) ON UPDATE
CASCADE ON DELETE CASCADE
);

-- Create the DriverShift table
CREATE TABLE DriverShift (
    NoOfDeliveries INT,
    ShiftNo INT,
    RecordId INT,
    StaffId CHAR(2),
    PRIMARY KEY (ShiftNo),
    FOREIGN KEY (ShiftNo) REFERENCES Shift(ShiftNo) ON DELETE NO ACTION ON
UPDATE NO ACTION,
    FOREIGN KEY (RecordId) REFERENCES DriverPay(RecordId) ON DELETE NO ACTION
ON UPDATE NO ACTION,
    FOREIGN KEY (StaffId) REFERENCES Driver(StaffId) ON DELETE NO ACTION ON
UPDATE NO ACTION
);

-- Create the DeliveryOrder table
```

```
CREATE TABLE DeliveryOrder (
    OrderId CHAR(5) PRIMARY KEY,
    DeliveryAddress VARCHAR(255),
    DeliveryTime DATETIME,
    ShiftNo INT,
    FOREIGN KEY (OrderId) REFERENCES PhoneOrder(OrderId) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (ShiftNo) REFERENCES DriverShift(ShiftNo) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Create the PickupOrder table
CREATE TABLE PickupOrder (
    OrderId CHAR(5) PRIMARY KEY,      -- Unique identifier for the pickup order
    DeliveryAddress VARCHAR(255),     -- Address for the pickup order
    DeliveryTime DATETIME,            -- Time for the pickup order
    FOREIGN KEY (OrderId) REFERENCES PhoneOrder (OrderId) ON UPDATE CASCADE ON DELETE CASCADE
);

--InStorePay table for staff payments related to in-store work
CREATE TABLE InStorePay (
    PaidHourlyRate DECIMAL(10, 2), -- The hourly rate paid to the staff
    HoursPaid DECIMAL(5, 2),      -- The number of hours for which the staff is paid
    RecordId INT PRIMARY KEY,      -- Unique identifier for the payment record
    FOREIGN KEY (RecordId) REFERENCES StaffPayment (RecordId) ON UPDATE CASCADE ON DELETE CASCADE
);

--InShopShift table for recording in-store staff shifts
CREATE TABLE InShopShift (
    NoOfHours DECIMAL(5, 2),      -- The number of hours worked in the shift
```

```
    ShiftNo INT,                -- Unique identifier for the shift
    RecordId INT,               -- Identifier for the related in-store payment record
    StaffId CHAR(2),            -- Staff identifier
    PRIMARY KEY (ShiftNo),
    FOREIGN KEY (ShiftNo) REFERENCES Shift (ShiftNo) ON DELETE NO ACTION ON
UPDATE NO ACTION,
    FOREIGN KEY (RecordId) REFERENCES InStorePay (RecordId) ON DELETE NO ACTION
ON UPDATE NO ACTION,
    FOREIGN KEY (StaffId) REFERENCES InStore (StaffId) ON DELETE NO ACTION ON
UPDATE NO ACTION
);

--IngredientOrder table for recording ingredient orders
CREATE TABLE IngredientOrder (
    IngredientOrderNo CHAR(10) PRIMARY KEY, -- Unique identifier for the ingredient order
    DateIssued DATE,                    -- Date when the order was issued
    DateSupplied DATE,                  -- Date when the ingredients were supplied
    Total DECIMAL(10, 2),               -- Total cost of the order
    Status VARCHAR(50),                 -- Status of the order
    Description TEXT,                   -- Description of the order
    SupplierID CHAR(5),                 -- Identifier for the supplier providing the ingredients
    FOREIGN KEY (SupplierID) REFERENCES Supplier (SupplierID) ON UPDATE CASCADE
ON DELETE CASCADE
);

--QuantityOrderMenuItem table for recording quantity ordered for menu items
CREATE TABLE QOrderMenuItem (
    ItemCode CHAR(5) PRIMARY KEY,  -- Identifier for the menu item and primary key
    OrderDate DATE,                -- Date of the order
    QuantityOrdered INT,           -- Quantity of the menu item ordered
    OrderId CHAR(5),               -- Identifier for the overall order
```

```
    FOREIGN KEY (ItemCode) REFERENCES MenuItem (ItemCode) ON UPDATE CASCADE
ON DELETE CASCADE,

    FOREIGN KEY (OrderId) REFERENCES Orders (OrderId) ON UPDATE CASCADE ON
DELETE CASCADE

);


-- Define the QuantityMenuItemIngredient table for recording quantities of ingredients used in
menu items

CREATE TABLE QMenuItemIngredient (

    ItemCode CHAR(5) PRIMARY KEY,    -- Identifier for the menu item and primary key

    IngredientCode CHAR(5),          -- Identifier for the ingredient

    QuantityUsed DECIMAL(10, 2),     -- Quantity of the ingredient used

    FOREIGN KEY (ItemCode) REFERENCES MenuItem (ItemCode) ON UPDATE CASCADE
ON DELETE CASCADE,

    FOREIGN KEY (IngredientCode) REFERENCES Ingredient (IngredientCode) ON UPDATE
CASCADE ON DELETE CASCADE

);


--QuantityIngredientIngOrder table for recording quantities of ingredients ordered in ingredient
orders

CREATE TABLE QIngredientIngOrder (

    IngredientOrderNo CHAR(10) PRIMARY KEY,  -- Identifier for the ingredient order and
primary key

    IngredientCode CHAR(5),              -- Identifier for the ingredient

    QuantityOrdered DECIMAL(10, 2),      -- Quantity of the ingredient ordered

    FOREIGN KEY (IngredientOrderNo) REFERENCES IngredientOrder (IngredientOrderNo)
ON UPDATE CASCADE ON DELETE CASCADE,

    FOREIGN KEY (IngredientCode) REFERENCES Ingredient (IngredientCode) ON UPDATE
CASCADE ON DELETE CASCADE

);


--QuantityIngredientSupplier table for recording quantities of ingredients supplied by suppliers

CREATE TABLE QIngredientSupplier (
```

```
    IngredientOrderNo CHAR(10) PRIMARY KEY, -- Identifier for the ingredient order and primary key
    SupplierID CHAR(5),                 -- Identifier for the supplier
    PricePerUnit DECIMAL(10, 2),          -- Price per unit of the ingredient
    FOREIGN KEY (SupplierID) REFERENCES Supplier (SupplierID) ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY (IngredientOrderNo) REFERENCES IngredientOrder (IngredientOrderNo) ON DELETE NO ACTION ON UPDATE NO ACTION
);



--Data Insertion starts here


-- Inserting data into the Customer table
INSERT INTO Customer (CustomerId, FirstName, Surname, Phone, Address)
VALUES
    ('111', 'John', 'Doe', '555-123-4567', '123 Main Street'),
    ('112', 'Jane', 'Smith', '555-987-6543', '456 Elm Avenue'),
    ('113', 'Bob', 'Johnson', '555-555-5555', '789 Oak Road'),
        ('114', 'Sarah', 'Johnson', '555-777-8888', '321 Pine Lane'),
    ('115', 'Michael', 'Brown', '555-444-3333', '555 Cedar Street'),
        ('116', 'David', 'Williams', '555-222-1111', '999 Maple Avenue'),
    ('117', 'Linda', 'Martinez', '555-333-2222', '123 Elm Street'),
    ('118', 'William', 'Anderson', '555-888-7777', '456 Oak Lane'),
    ('119', 'Emily', 'Taylor', '555-999-8888', '789 Cedar Road'),
    ('120', 'Richard', 'Moore', '555-666-5555', '234 Birch Street');


--Inserting data into the Staff table
INSERT INTO Staff (StaffId, TaxFileNo, FirstName, Surname, Phone, Description, Address, Status, AcName, BSB, AcNo)
VALUES
```

('11', '123-45-6789', 'John', 'Doe', '555-123-4567', 'Manager', '123 Main St, City', 'Active', 'John Doe', '123456', '12345678'),

('12', '987-65-4321', 'Jane', 'Smith', '555-234-5678', 'Crew Member', '456 Elm St, Town', 'Active', 'Jane Smith', '987654', '87654321'),

('13', '456-78-9012', 'Robert', 'Johnson', '555-345-6789', 'Cashier', '789 Oak St, Village', 'Inactive', 'Robert Johnson', '456789', '98761234'),

('14', '789-01-2345', 'Lisa', 'Wilson', '555-456-7890', 'Manager', '101 Pine St, County', 'Active', 'Lisa Wilson', '789012', '23456789'),

('15', '234-56-7890', 'Michael', 'Brown', '555-567-8901', 'Crew Member', '202 Cedar St, Borough', 'Inactive', 'Michael Brown', '234567', '89012345'),

('16', '345-67-8901', 'Emily', 'Davis', '555-678-9012', 'Cashier', '303 Birch St, Township', 'Active', 'Emily Davis', '345678', '90123456'),

('17', '567-89-0123', 'David', 'Jones', '555-789-0123', 'Manager', '404 Maple St, District', 'Active', 'David Jones', '567890', '01234567'),

('18', '678-90-1234', 'Sarah', 'Lee', '555-890-1234', 'Crew Member', '505 Willow St, Region', 'Inactive', 'Sarah Lee', '678901', '12345678'),

('19', '890-12-3456', 'James', 'Wilson', '555-901-2345', 'Cashier', '606 Cedar St, Province', 'Active', 'James Wilson', '890123', '23456789'),

('20', '012-34-5678', 'Laura', 'Smith', '555-012-3456', 'Manager', '707 Oak St, State', 'Active', 'Laura Smith', '012345', '34567890');

-- Insert data into the MenuItem table

INSERT INTO MenuItem (ItemCode, Name, Size, Price, Description)

VALUES

('M0001', 'Hamburger', 'Regular', 5.99, 'Classic beef burger'),

('M0002', 'Cheeseburger', 'Regular', 6.49, 'Burger with cheese'),

('M0003', 'Chicken Sandwich', 'Regular', 6.99, 'Grilled chicken sandwich'),

('M0004', 'Fries', 'Regular', 2.99, 'Crispy potato fries'),

('M0005', 'Soda', 'Small', 1.99, 'Carbonated soft drink'),

('M0006', 'Pizza Slice', 'Large', 4.99, 'Delicious pizza slice'),

('M0007', 'Salad', 'Regular', 4.49, 'Fresh garden salad'),

('M0008', 'Pasta', 'Large', 7.99, 'Homemade pasta dish'),

('M0009', 'Ice Cream', 'Small', 3.99, 'Sweet vanilla ice cream'),

('M0010', 'Smoothie', 'Medium', 5.49, 'Refreshing fruit smoothie');

```
-- Insert data into the Supplier table

INSERT INTO Supplier (SupplierID, Name, Phone, Address, eMail, ContactPerson)

VALUES

    ('S0001', 'Fresh Food Inc.', '555-111-1111', '123 Supplier St', 'info@freshfood.com', 'John Supplier'),

    ('S0002', 'Bulk Ingredients Ltd.', '555-222-2222', '456 Distributor Dr', 'info@bulkingredients.com', 'Alice Distributor'),

    ('S0003', 'Farm to Table Produce', '555-333-3333', '789 Farm Rd', 'info@farmtotable.com', 'Bob Farmer'),

    ('S0004', 'Beverage World', '555-444-4444', '101 Beverage Ave', 'info@beverageworld.com', 'Laura Beverage'),

    ('S0005', 'Meat Master', '555-555-5555', '246 Meat Blvd', 'info@meatmaster.com', 'Mark Meat'),

    ('S0006', 'Fresh Veggies Co.', '555-666-6666', '789 Veggie Ln', 'info@freshveggies.com', 'Sarah Veggie'),

    ('S0007', 'Dairy Delights', '555-777-7777', '456 Dairy Rd', 'info@dairydelights.com', 'David Dairy'),

    ('S0008', 'Seafood Sensations', '555-888-8888', '101 Seafood St', 'info@seafoodsensations.com', 'Samantha Seafood'),

    ('S0009', 'Bakery Bliss', '555-999-9999', '246 Bakery Ave', 'info@bakerybliss.com', 'Brian Baker'),

    ('S0010', 'Spice Emporium', '555-101-1010', '123 Spice Rd', 'info@spiceemporium.com', 'Linda Spice');


-- Insert dummy data into the Ingredient table

INSERT INTO Ingredient (IngredientCode, Name, StockUnit, Description, StockLevelAtStockTake, DateOfLastStockTake, SuggestedStockLevel, ReorderLevel, Type)

VALUES

    ('In001', 'Flour', 'Kilograms', 'All-purpose flour', 500.00, '2023-10-15', 200.00, 100.00, 'Dry Goods'),

    ('In002', 'Tomatoes', 'Pounds', 'Fresh red tomatoes', 150.50, '2023-10-15', 50.00, 30.00, 'Produce'),

    ('In003', 'Chicken', 'Pounds', 'Boneless chicken breast', 100.25, '2023-10-15', 40.00, 20.00, 'Meat'),
```

```
    ('In004', 'Milk', 'Liters', 'Whole milk', 200.75, '2023-10-15', 80.00, 50.00, 'Dairy'),
    ('In005', 'Lettuce', 'Pounds', 'Fresh lettuce', 50.25, '2023-10-15', 20.00, 10.00, 'Produce'),
    ('In006', 'Pasta', 'Kilograms', 'Spaghetti pasta', 60.00, '2023-10-15', 30.00, 15.00, 'Dry
Goods'),
    ('In007', 'Onions', 'Pounds', 'Yellow onions', 40.50, '2023-10-15', 15.00, 10.00, 'Produce'),
    ('In008', 'Cheese', 'Kilograms', 'Cheddar cheese', 80.25, '2023-10-15', 30.00, 20.00, 'Dairy'),
    ('In009', 'Beef', 'Pounds', 'Ground beef', 90.75, '2023-10-15', 40.00, 30.00, 'Meat'),
    ('In010', 'Rice', 'Kilograms', 'Long-grain rice', 75.00, '2023-10-15', 25.00, 15.00, 'Dry Goods');

-- Inserting data into the InStore table
INSERT INTO InStore (StaffId, HourlyRate, ShiftNo)
VALUES
    ('11', 10.50, 1),
    ('12', 9.75, 2),
    ('13', 11.00, 3),
    ('14', 10.25, 4),
    ('15', 12.00, 5);

-- Inserting data into the Driver table
INSERT INTO Driver (StaffId, DriverLicNo, RatePerDelivery, ShiftNo)
VALUES
    ('16', 123456, 5.50, 6),
    ('17', 789012, 6.25, 7),
    ('18', 345678, 5.75, 8),
    ('19', 901234, 6.00, 9),
    ('20', 567890, 7.00, 10);

-- Inserting data into the Shift table
INSERT INTO Shift (ShiftNo, StartDateTime, HoursPaid)
```

```
VALUES
    (1, '2023-10-25 08:00:00', 8.5),
    (2, '2023-10-25 09:30:00', 7.25),
    (3, '2023-10-25 12:00:00', 6.75),
    (4, '2023-10-25 14:30:00', 7.0),
    (5, '2023-10-26 07:00:00', 8.0),
    (6, '2023-10-26 08:30:00', 6.5),
    (7, '2023-10-26 11:00:00', 7.75),
    (8, '2023-10-26 14:00:00', 7.25),
    (9, '2023-10-27 09:00:00', 7.5),
    (10, '2023-10-29 14:00:00', 7.25);


-- Inserting data into the Orders table
INSERT INTO Orders (OrderId, OrderDateTime, OrderType, TotalAmountDue,
PaymentMethod, PaymentApprovalNo, Status, CustomerId, StaffId)
VALUES
    ('ORD01', '2023-10-25 14:30:00', 'Online', 100.50, 'Credit Card', '123456789', 'Processing',
'111', '11'),
    ('ORD02', '2023-10-26 10:45:00', 'In-Store', 75.25, 'Cash', '987654321', 'Completed', '112',
'12'),
    ('ORD03', '2023-10-27 16:15:00', 'Online', 150.75, 'PayPal', '456789123', 'Processing', '113',
'13'),
    ('ORD04', '2023-10-28 12:20:00', 'In-Store', 45.99, 'Credit Card', '789123456', 'Completed',
'114', '14'),
    ('ORD05', '2023-10-29 09:00:00', 'Online', 200.00, 'PayPal', '567891234', 'Processing', '115',
'15'),
        ('ORD06', '2023-10-30 15:45:00', 'Online', 80.99, 'Credit Card', '234567890',
'Processing', '116', '11'),
    ('ORD07', '2023-10-31 11:10:00', 'In-Store', 60.75, 'Cash', '876543210', 'Completed', '117',
'12'),
    ('ORD08', '2023-11-01 13:25:00', 'Online', 125.50, 'PayPal', '678901234', 'Processing', '118',
'13'),
    ('ORD09', '2023-11-02 08:30:00', 'In-Store', 95.25, 'Credit Card', '345678901', 'Completed',
'119', '14'),
```

```
    ('ORD10', '2023-11-03 17:20:00', 'Online', 175.00, 'PayPal', '789012345', 'Processing', '120',
'15');


-- Insert data into the WalkInOrder table
INSERT INTO WalkInOrder (OrderId, WalkInTime)
VALUES
    ('ORD01', '2023-10-25 14:30:00'),
    ('ORD02', '2023-10-26 10:45:00'),
    ('ORD03', '2023-10-27 16:15:00'),
    ('ORD04', '2023-10-28 12:20:00'),
    ('ORD05', '2023-10-29 09:00:00');


-- Insert data into the PhoneOrder table
INSERT INTO PhoneOrder (OrderId, TimeCallAnswered, TimeCallTerminated)
VALUES
    ('ORD06', '2023-10-25 14:30:00', '2023-10-25 14:45:00'),
    ('ORD07', '2023-10-26 10:45:00', '2023-10-26 11:10:00'),
    ('ORD08', '2023-10-27 16:15:00', '2023-10-27 16:30:00'),
    ('ORD09', '2023-10-28 12:20:00', '2023-10-28 12:40:00'),
    ('ORD10', '2023-10-29 09:00:00', '2023-10-29 09:20:00');


-- Insert data into the StaffPayment table
INSERT INTO StaffPayment (RecordId, GrossPay, TaxWithHeld, TotalAmountPaid,
PaymentDate, PayPeriodStartDate, PayPeriodEndDate, StaffId)
VALUES
    (1, 1500.00, 300.00, 1200.00, '2023-10-01', '2023-09-16', '2023-09-30', '11'),
    (2, 1400.00, 280.00, 1120.00, '2023-10-01', '2023-09-16', '2023-09-30', '12'),
    (3, 1600.00, 320.00, 1280.00, '2023-10-01', '2023-09-16', '2023-09-30', '13'),
    (4, 1450.00, 290.00, 1160.00, '2023-10-01', '2023-09-16', '2023-09-30', '14'),
    (5, 1550.00, 310.00, 1240.00, '2023-10-01', '2023-09-16', '2023-09-30', '15'),
        (6, 1500.00, 300.00, 1200.00, '2023-11-01', '2023-10-01', '2023-10-15', '16'),
```

```
    (7, 1400.00, 280.00, 1120.00, '2023-11-01', '2023-10-01', '2023-10-15', '17'),
    (8, 1600.00, 320.00, 1280.00, '2023-11-01', '2023-10-01', '2023-10-15', '18'),
    (9, 1450.00, 290.00, 1160.00, '2023-11-01', '2023-10-01', '2023-10-15', '18'),
    (10, 1550.00, 310.00, 1240.00, '2023-11-01', '2023-10-01', '2023-10-15', '20');

-- Insert data into the DriverPay table
INSERT INTO DriverPay (PaidDeliveryRate, DeliveriesPaid, RecordId)
VALUES
    (5.00, 10, 1),
    (5.50, 12, 2),
    (4.75, 8, 3),
    (6.00, 14, 4),
    (5.25, 11, 5),
    (5.75, 13, 6),
    (4.50, 9, 7),
    (6.25, 15, 8),
    (5.00, 10, 9),
    (6.50, 16, 10);

-- Inserting data into the DriverShift table
INSERT INTO DriverShift (NoOfDeliveries, ShiftNo, RecordId, StaffId)
VALUES
    (10, 1, 1, '16'),
    (8, 2, 2, '17'),
    (12, 3, 3, '18'),
    (5, 4, 4, '19'),
    (9, 5, 5, '20');

-- Inserting data into the DeliveryOrder table
INSERT INTO DeliveryOrder (OrderId, DeliveryAddress, DeliveryTime, ShiftNo)
```

```
VALUES
    ('ORD06', '123 Main St, Cityville', '2023-10-26 14:00:00', 1),
    ('ORD07', '456 Elm St, Townsville', '2023-10-26 15:30:00', 2),
    ('ORD08', '789 Oak St, Villagetown', '2023-10-26 16:45:00', 3),
    ('ORD09', '101 Pine St, Hamletville', '2023-10-26 17:15:00', 4),
    ('ORD10', '202 Maple St, Boroughtown', '2023-10-26 18:30:00', 5);

-- Insert data into the PickupOrder table
INSERT INTO PickupOrder (OrderId, DeliveryAddress, DeliveryTime)
VALUES
        ('ORD06', '123 Main St, City, State', '2023-10-26 14:00:00'),
    ('ORD07', '456 Elm St, City, State', '2023-10-27 10:30:00'),
    ('ORD08', '789 Oak St, City, State', '2023-10-28 16:45:00'),
    ('ORD09', '321 Pine St, City, State', '2023-10-29 12:15:00'),
    ('ORD10', '654 Birch St, City, State', '2023-10-30 18:00:00');

-- Insert data into the InStorePay table
INSERT INTO InStorePay (PaidHourlyRate, HoursPaid, RecordId)
VALUES
        (15.00, 40.5, 1),
    (14.50, 37.0, 2),
    (16.00, 45.5, 3),
    (13.75, 32.0, 4),
    (15.25, 38.5, 5);

-- Insert 5 rows of data into the table
INSERT INTO InShopShift (NoOfHours, ShiftNo, RecordId, StaffId)
VALUES (8.5, 1, 1, '11'),
    (7.0, 2, 2, '12'),
    (9.0, 3, 3, '13'),
```

```
    (6.5, 4, 4, '14'),
    (7.5, 5, 5, '15');

-- Inserting data into the table
INSERT INTO IngredientOrder (IngredientOrderNo, DateIssued, DateSupplied, Total, Status, Description, SupplierID)
VALUES
    ('IO001', '2023-10-01', '2023-10-05', 500.00, 'Delivered', 'First ingredient order', 'S0001'),
    ('IO002', '2023-10-03', '2023-10-06', 750.00, 'Delivered', 'Second ingredient order', 'S0002'),
    ('IO003', '2023-10-05', '2023-10-08', 600.00, 'Delivered', 'Third ingredient order', 'S0003'),
    ('IO004', '2023-10-08', '2023-10-10', 450.00, 'Delivered', 'Fourth ingredient order', 'S0004'),
    ('IO005', '2023-10-10', '2023-10-14', 800.00, 'In Progress', 'Fifth ingredient order', 'S0005'),
    ('IO006', '2023-10-12', '2023-10-16', 550.00, 'In Progress', 'Sixth ingredient order', 'S0006'),
    ('IO007', '2023-10-15', '2023-10-18', 700.00, 'In Progress', 'Seventh ingredient order', 'S0007'),
    ('IO008', '2023-10-18', '2023-10-21', 900.00, 'Ordered', 'Eighth ingredient order', 'S0008'),
    ('IO009', '2023-10-20', '2023-10-24', 600.00, 'Ordered', 'Ninth ingredient order', 'S0009'),
    ('IO010', '2023-10-23', '2023-10-27', 750.00, 'Ordered', 'Tenth ingredient order', 'S0010');

-- Insert data into the table
INSERT INTO QOrderMenuItem (ItemCode, OrderDate, QuantityOrdered, OrderId)
VALUES
    ('M0001', '2022-10-01', 3, 'ORD01'),
    ('M0002', '2023-10-01', 2, 'ORD02'),
    ('M0003', '2021-10-01', 4, 'ORD03'),
    ('M0004', '2023-10-02', 1, 'ORD04'),
    ('M0005', '2022-10-02', 5, 'ORD05'),
    ('M0006', '2023-10-03', 2, 'ORD06'),
    ('M0007', '2021-10-03', 3, 'ORD07'),
    ('M0008', '2023-10-03', 2, 'ORD08'),
    ('M0009', '2024-10-04', 1, 'ORD09'),
```

```
    ('M0010', '2023-10-04', 3, 'ORD10');

-- Insert data into the QMenuItemIngredient table
INSERT INTO QMenuItemIngredient (ItemCode, IngredientCode, QuantityUsed)
VALUES
    ('M0001', 'In001', 2.5),
    ('M0002', 'In002', 1.0),
    ('M0003', 'In003', 1.0),
    ('M0004', 'In004', 0.5),
    ('M0005', 'In005', 3.0),
    ('M0006', 'In006', 1.5),
    ('M0007', 'In007', 2.0),
    ('M0008', 'In008', 1.0),
    ('M0009', 'In009', 2.5),
    ('M0010', 'In010', 1.5);

-- Insert data into the QIngredientIngOrder table
INSERT INTO QIngredientIngOrder (IngredientOrderNo, IngredientCode, QuantityOrdered)
VALUES
    ('IO001', 'In001', 20.0),
    ('IO002', 'In002', 10.0),
    ('IO003', 'In003', 15.0),
    ('IO004', 'In004', 30.0),
    ('IO005', 'In005', 5.0),
    ('IO006', 'In006', 12.0),
    ('IO007', 'In007', 25.0),
    ('IO008', 'In008', 18.0),
    ('IO009', 'In009', 8.0),
    ('IO010', 'In010', 11.0);
```

```
-- Insert data into the QIngredientSupplier table
INSERT INTO QIngredientSupplier (IngredientOrderNo, SupplierID, PricePerUnit)
VALUES
    ('IO001', 'S0001', 2.5),
    ('IO002', 'S0002', 3.0),
    ('IO003', 'S0003', 2.2),
    ('IO004', 'S0004', 2.6),
    ('IO005', 'S0005', 1.8),
    ('IO006', 'S0006', 2.0),
    ('IO007', 'S0007', 1.5),
    ('IO008', 'S0008', 1.9),
    ('IO009', 'S0009', 3.2),
    ('IO010', 'S0010', 2.8);

--Q1
SELECT FirstName, Surname, HourlyRate
FROM Staff
JOIN InStore ON InStore.StaffId = Staff.StaffId
WHERE Staff.StaffId = '11';

--Q2
SELECT Shift.ShiftNo, Shift.StartDateTime, Shift.HoursPaid
FROM Staff
JOIN DriverShift ON Staff.StaffId = DriverShift.StaffId
JOIN Shift ON DriverShift.ShiftNo = Shift.ShiftNo
WHERE Staff.FirstName = 'James'
  AND Staff.Surname = 'Wilson'
  AND Shift.StartDateTime BETWEEN '2023-10-26 14:00:00' AND '2029-10-26 14:00:00';

--Q3
```

```
SELECT O.OrderId, O.OrderDateTime, O.OrderType, O.TotalAmountDue, O.PaymentMethod,
O.PaymentApprovalNo, O.Status, C.FirstName, C.Surname

FROM Orders O

JOIN Customer C ON O.CustomerId = C.CustomerId

JOIN WalkInOrder WIO ON O.OrderId = WIO.OrderId

WHERE C.FirstName = 'John'

  AND C.Surname = 'Doe'

  AND O.OrderDateTime BETWEEN '2023-10-26 10:45:00' AND '2023-11-02 08:30:00'

  AND O.OrderType = 'In-Store';


--Q4

SELECT DISTINCT M.Name

FROM MenuItem M

JOIN QOrderMenuItem Q ON M.ItemCode = Q.ItemCode

WHERE YEAR(Q.OrderDate) = YEAR('2023');


--Q5
```

Summary

The Number One Pizza database project has evolved from its initial conceptual model, based on requirements analysis, to a fully realized and implemented relational database. The database now incorporates SQL scripts that define the tables and relationships, enhancing data integrity and efficiency. This database encompasses a wide range of data related to customers, employees, suppliers, orders, menus, ingredients, and more, serving as a central repository for critical business information. The project has established clear business regulations and transaction handling methods, optimizing order processing, menu management, ingredient tracking, and employee management. These rules are designed to streamline operations and maintain efficient workflows within the organization. The Entity-Relationship Diagram (ERD) visually represents the complexity of the database, illustrating intricate data connections and essential data flows. With the inclusion of SQL scripts, the project has progressed from a conceptual model to a relational model in DBML format. This model has undergone normalization to achieve Boyce-Codd Normal Form (BCNF), ensuring data integrity and efficiency at its core.

In conclusion, the Number One Pizza database project has successfully met the database requirements, providing the organization with a robust data management system to enhance and optimize their business operations.