Table of Contents

Popular Pizza Requirements

Data Requirements

Customer Order

Popular Pizza takes orders via phone as well as in store "walk-in" customers. They offer both pick up in store and delivery services. The data required for every order includes the order number, customer ID (If the order is placed in store the customer's name is recorded to identify the order), the order type (in store or via phone), the delivery method (pickup in store or delivery), the discount code (if applicable), payment method (credit card, debit card or cash), order total price, tax paid and order status (processing, with delivery driver, delivered etc..) along with the payment approval number if customer payed with card, order verification status if ordered via phone (order is valid or a hoax) along with the verification call start and end time.

Customer

If a customer orders via phone their customer ID, phone number, name (first name & last name) and postal address are recorded in the database.

Discount Program

Popular Pizza offers different types of discount programs. The discount code, program start and end date, discount requirements, discount percentage and description are recorded in the database. Discounts are only applied to the order total.

Staff

Popular Pizza's employees are divided into two types: Those who work in store (paid hourly) and those who carry out deliveries (paid per delivery). For every employee, there is an employee number, name (first name & last name), postal address, contact number, bank details (account number, BSB number and bank name), employment status, description and payment rate (per hour or per delivery). For employees that work as drivers their drivers licence number is also recorded.

Shift

All employees shift details (start date, start time, end date and end time) is recorded in the database along with the shift number. If an employee is a driver the number of orders they delivered is also recorded. If the employee works in-store his/her hours worked is recorded.

Payment

Employee payments are made for each shift worked to the employees' bank account. The payment ID (receipt number), amount, shift starting date and date of payment is recorded in the database. Delivery Drivers are paid per delivery and In-store staff are paid per hour. Therefore, the number of deliveries made by a driver is also recorded.

Menu Item

For every menu item the item code (unique), name, description, size and current selling price are recorded in the database. Every menu item is made up of a number of ingredients.

Ingredient

Each ingredient has its own unique code, name, type, description, stock level at current stocktake period, date last stocktake was taken, stock level at the last stocktake, suggested current stock level, reorder level and the supplier number for the supplier who supplies the ingredient.

Supplier

For every one of Popular Pizza’s suppliers the supplier number, name, address, phone number and the name of the person to contact is recorded in the database.

Ingredient Order

Every time an ingredient order is placed with a supplier the order number, date ordered, date the order was received, total amount of ingredients, total price of order, tax paid, order status, description, supplier number and the ingredient code gets recorded in the database.

Transaction Requirements

Data Entry

Enter the details of a new staff member (0001, Delivery, John, Smith, 6, Lyle St, Ryde, NSW, 2112, 02 5550 2809, 865414088, 67154-784, ANZ, 012040, part-time, Cashier, 16.00, 103805501).

Enter shift details at end of shift (such as 56752, 0001, 14-08-2017, 14-08-2017, 09:00, 17:00, 50)

Enter payment details for a new employee (such as 7468842, 0001, 128.00, 56752, 14-05-2017).

Enter the details of a new customer (such as 0001, 02 5550 2809, John, Smith, 6, Lyle St, Ryde, NSW, 2112).

Enter customer order details (such as 75681, 21-08-2017, 0001, Phone, Pickup, 753475, Debit Card, 50.00, 05.00, Complete, 1234, valid, 16:47, 16:49).

Enter the details of a new supplier (such as 8251, Joe’s Meats, 10, Lyle St, Ryde, NSW, 2112, 025550 2810, Joe, Jones)

Enter the details for a new discount program (such as 753475, 21-08-2017, 25-08-2017, Buy two “Chef’s Special” pizzas, 25, Buy two “Chef’s Special” pizzas get 25% off.).

Enter a new menu item (such as 0001, Chicken & Camembert Pizza, Pizza with chicken & camembert cheese, Large, 12.50).

Enter a new ingredient (such as 0005, Chicken Breast, Meat, Chicken Breast, 500, 14-08-2017, 135, 600, 8251).

Enter the details for a new ingredient order (such as 84618, 11-08-2017, 14-08-2017, 465, 500.00, 50.00, Delivered, Order for chicken breasts, 8251, 0005).

Data update/deletion

Update/delete the details for an employee.

Update/delete the details for a customer.

Update/delete the details for ingredients.

Update/delete the details for ingredient orders.

Update/delete the details for suppliers.

Update/delete the details for menu items.

Update/delete the details for a discount program.

Update/delete an employee’s bank details.

Update/delete the details for a customer order.

Update/delete the details for an employee shift.

Update/delete the details for an employee payment.

Data Queries

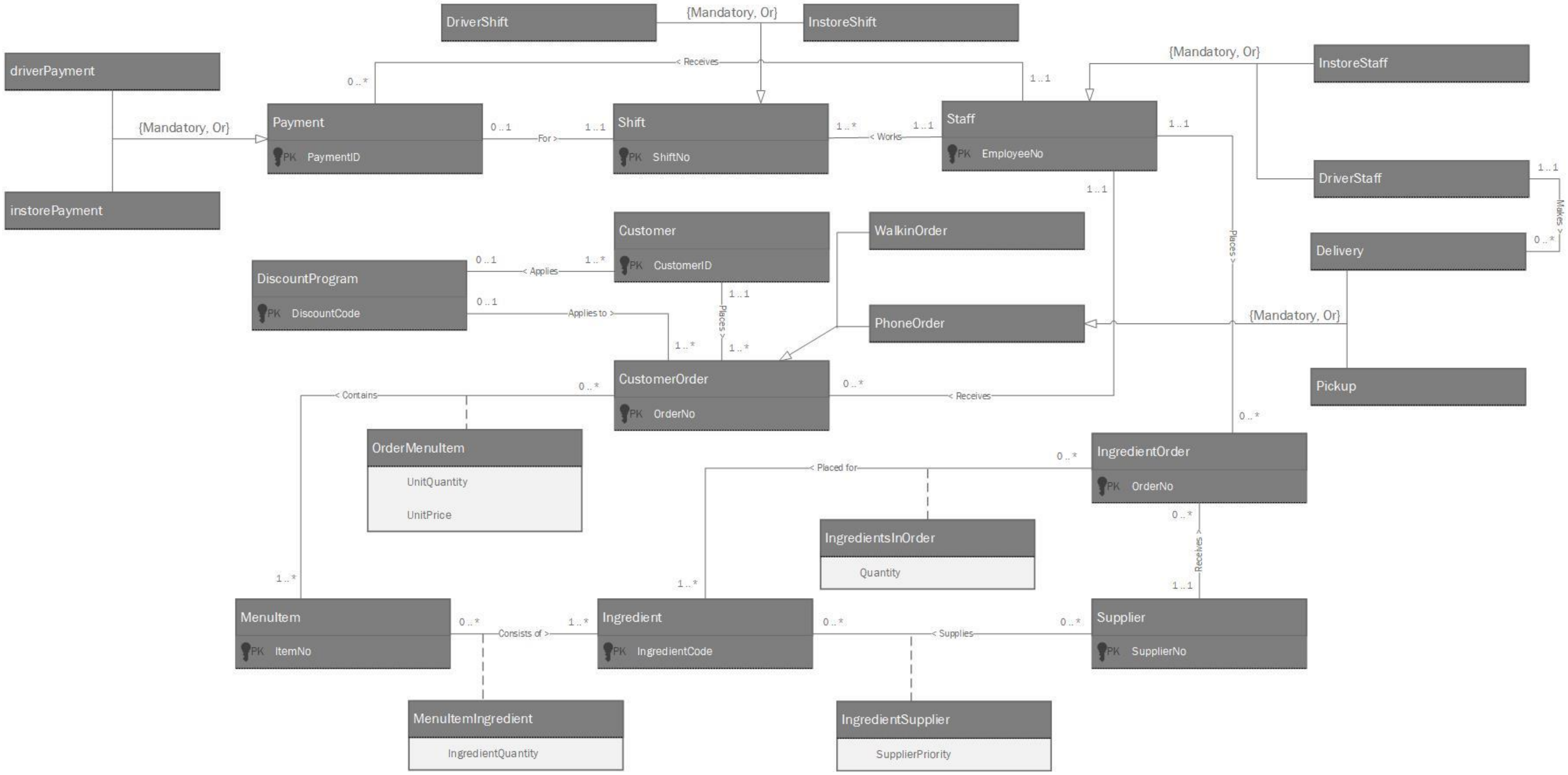
Examples of queries required by Popular Pizza’s employees.

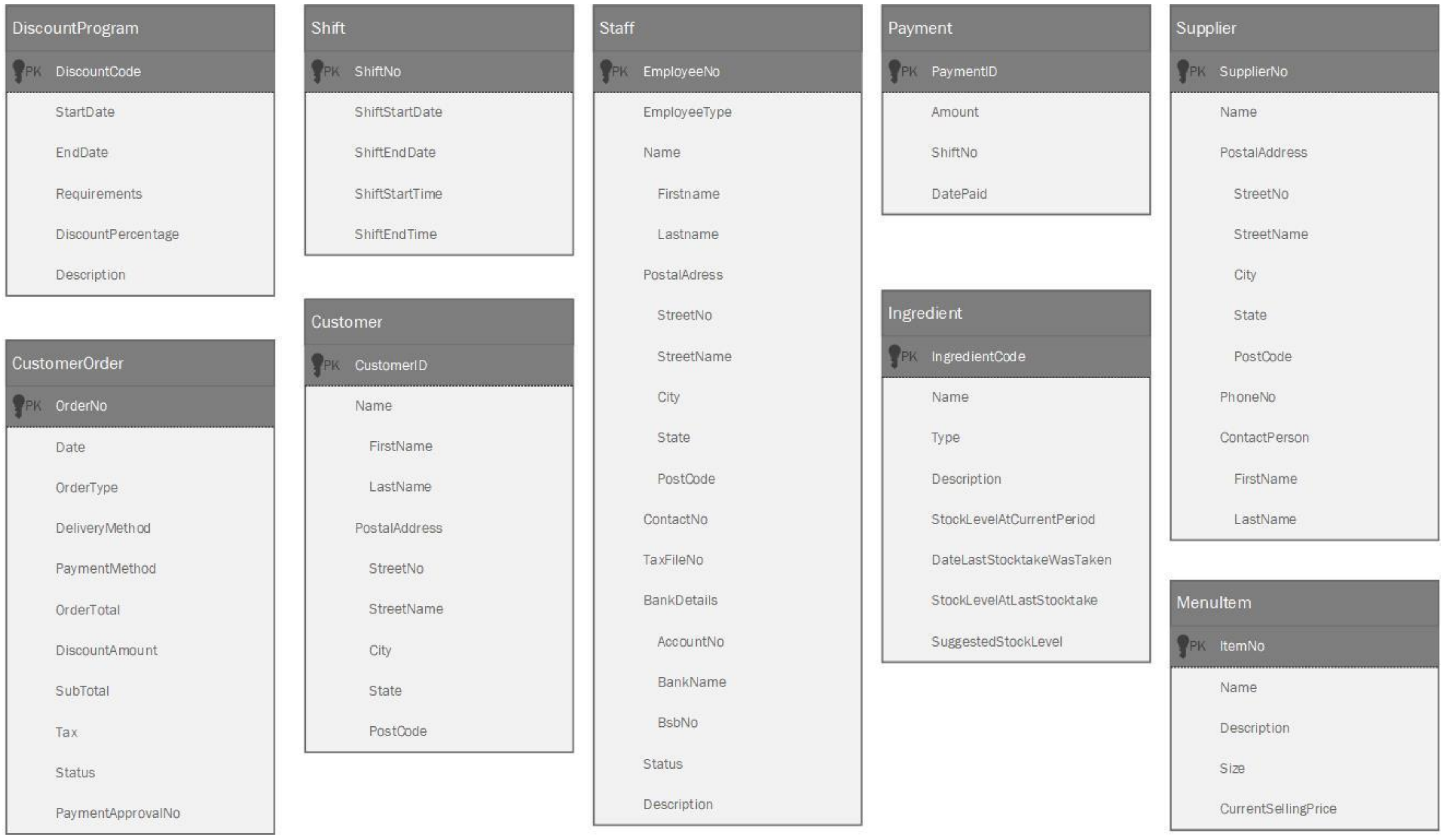
a) List a customer’s details.
b) Identify a customer order’s details.
c) List current discount program’s details.
d) List menu items.
e) List details about a particular ingredient.
f) List ingredient order details.
g) List their own shift details.
h) Identify their own employee information.
i) Identify their own bank details.
j) Identify their own payment information.
k) List suppliers for an ingredient.
l) Identify a supplier’s information

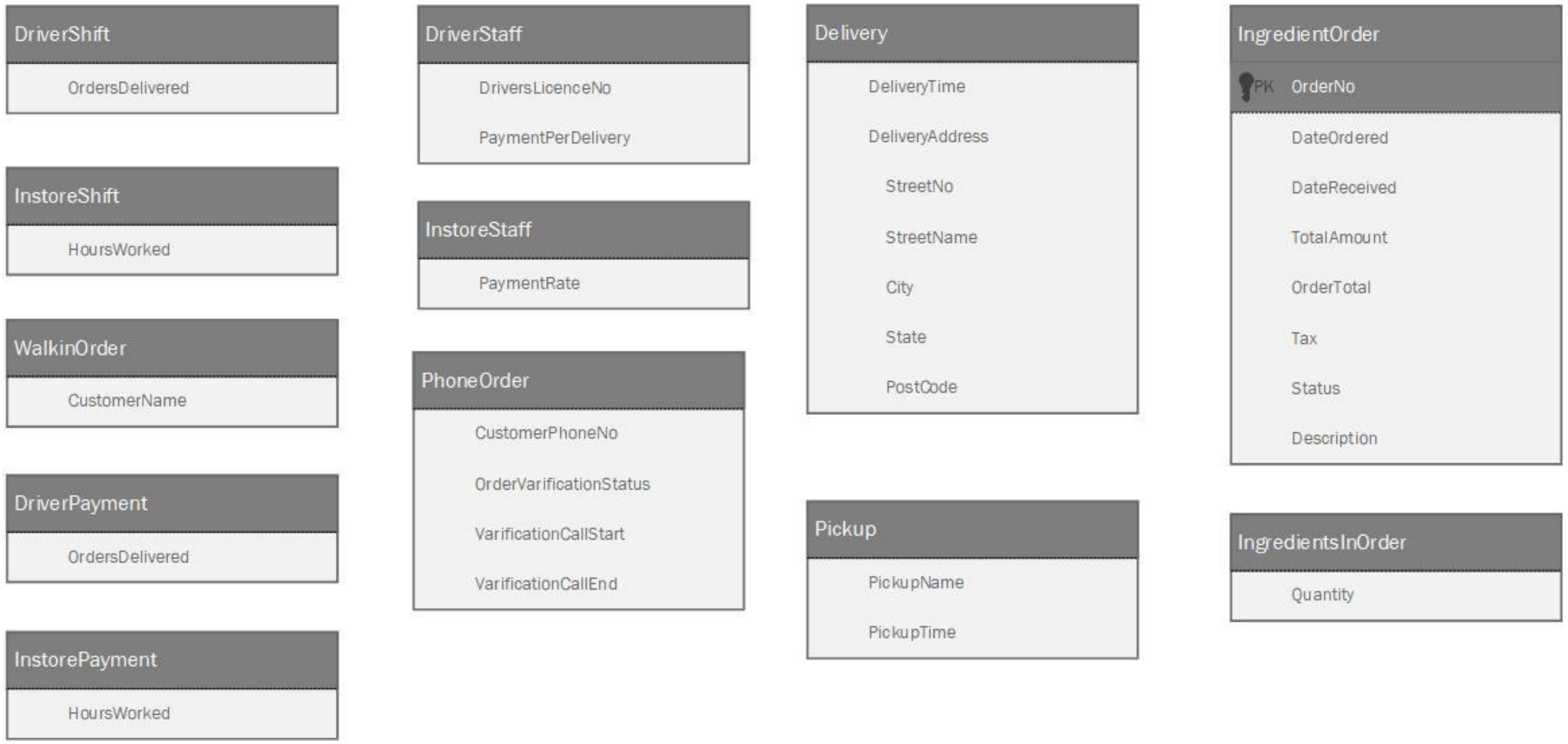
Business Rules

- The amount of each ingredient remaining must be updated every time some is used.
- The results of the weekly stocktake must be input into the database.
- When an ingredients stock level decreases below its reorder level an order for the ingredient must be placed.
- A new customer must be marked as un-verified until the verification process is successfully completed.
    - An order placed over the phone will only be valid once it has been verified and has a 'verified' status.
    - If the name and address a customer provides does not match an existing record a new record must be created and the verification process must take place.
- Employees must record each shift they work in the database.
- An employee can only be either an in-store worker or a delivery driver.
- Employees cannot delete data from the database.
- An Employees' status can only be either:
    - Full time
    - Part time
- Payments can only be added by accounting staff
- An orders payment method can only have one of the following values:
    - Credit card
    - Debit card
    - Cash
- An orders type can only be either:
    - Pick up
    - Delivery
- If an order is paid for using a card the approval number must be stored in the order's paymentApprovalNo.
- Only one discount program can be used on an order.
- Discount Programs can only be added/updated by an administrator

EER Model & Data Dictionary




DiscountProgram
PK DiscountCode
StartDate
EndDate
Requirements
DiscountPercentage
Description
CustomerOrder
PK OrderNo
Date
OrderType
DeliveryMethod
PaymentMethod
OrderTotal
DiscountAmount
SubTotal
Tax
Status
PaymentApprovalNo
Shift
PK ShiftNo
ShiftStartDate
ShiftEndDate
ShiftStartTime
ShiftEndTime
Customer
PK CustomerID
Name
FirstName
LastName
PostalAddress
StreetNo
StreetName
City
State
PostCode
Staff
PK EmployeeNo
EmployeeType
Name
Firstname
Lastname
PostalAdress
StreetNo
StreetName
City
State
PostCode
ContactNo
TaxFileNo
BankDetails
AccountNo
BankName
BsbNo
Status
Description
Payment
PK PaymentID
Amount
ShiftNo
DatePaid
Ingredient
PK IngredientCode
Name
Type
Description
StockLevelAtCurrentPeriod
DateLastStocktakeWasTaken
StockLevelAtLastStocktake
SuggestedStockLevel
Supplier
PK SupplierNo
Name
PostalAddress
StreetNo
StreetName
City
State
PostCode
PhoneNo
ContactPerson
FirstName
LastName
MenuItem
PK ItemNo
Name
Description
Size
CurrentSellingPrice


DriverShift
OrdersDelivered
InstoreShift
HoursWorked
WalkinOrder
CustomerName
DriverPayment
OrdersDelivered
InstorePayment
HoursWorked
DriverStaff
DriversLicenceNo
PaymentPerDelivery
InstoreStaff
PaymentRate
PhoneOrder
CustomerPhoneNo
OrderVarificationStatus
VarificationCallStart
VarificationCallEnd
Delivery
DeliveryTime
DeliveryAddress
StreetNo
StreetName
City
State
PostCode
Pickup
PickupName
PickupTime
IngredientOrder
PK OrderNo
DateOrdered
DateReceived
TotalAmount
OrderTotal
Tax
Status
Description
IngredientsInOrder
Quantity

Entity Types

Entity Name	Description	Aliases	Occurrence
Staff	General term describing all staff employed by Popular Pizza	Employee	Every staff member works one or more shifts. Staff may receive many customer orders Staff may place many ingredient orders. Staff members receive payments per shift worked
DriverStaff	General term describing staff responsible for pizza deliveries.		Staff members may be delivery drivers.
InstoreStaff	General term describing staff that work in-store.		Staff members may work in-store.
Shift	General term describing shifts worked by staff members		One or more shifts are worked by a staff member. For every shift worked a staff member receives a payment.
DriverShift	General term describing shifts worked by delivery drivers.		A shift may be worked by a delivery driver.
InstoreShift	General term describing shifts worked by in-store staff.		A shift may be worked by an in-store staff member.
Payment	General term describing staff payments.		Staff members receive zero or more payments. A single payment is made for a single shift worked.
Customer	General term describing all customers who buy from Popular Pizza	Client	Customers can place one or more orders.
DriverPayment	General term describing payments received by driver staff.		Driver staff receive zero or more payments. for a shift worked.
InstorePayment	General term describing payments received by in-store staff.		In-store staff receive zero or more payments. for a shift worked.

Entity Name	Description	Aliases	Occurrence
CustomerOrder	General term describing a customer's order.	Order	A customer places one or more orders. Zero or more customer orders has one or more menu items. A staff member receives zero or more customer orders. A single discount program may apply to many customer orders.
WalkinOrder	General term describing a walk-in customer's order.		An order may be a walk-in order.
PhoneOrder	General term describing a phone customer's order.		An order may be a phone order.
Delivery	General term describing orders that are delivered.		A phone order may be delivered to the customer.
Pickup	General term describing orders that are picked up in store.		A phone order may be picked up in-store.
DiscountProgram	General term describing Popular Pizza's discount programs.		Zero or one discount program applies to one or more orders.
MenuItem	General term describing menu items at Popular Pizza.		Zero or more customer orders contain one or more menu items. Zero or more menu items consist of one or more ingredients.
OrderMenuItem	General term describing the quantity of a single menu item in an order.		A customer order can contain more than one identical menu item.
Ingredient	General term describing Popular Pizza's ingredients		One or more Menu items consist of one or more ingredients. One or more suppliers supply one or more ingredients. Zero or more ingredient orders are placed for one or more ingredients.

Entity Name	Description	Aliases	Occurrence
MenuItemIngredient	General term describing the quantity of a single ingredient in a menu item.		A menu item has a quantity for an ingredient.
IngredientOrder	General term describing an order for an ingredient.		A staff member places zero or more ingredient orders. Zero or more ingredient orders are placed for one or more ingredfients. A supplier may receive zero or more ingredient orders.
IngredientsInOrder	General term describing the ingredients in an ingredient order.		An ingredient order has a quantity for an ingredient.
Supplier	General term describing suppliers of ingredients.		A supplier may receive zero or more ingredient orders. Zero or more suppliers supplies zero or more ingredients.
IngredientSupplier	General term describing the association between ingredients and suppliers.		An ingredient may have multiple suppliers and a supplier may supply multiple ingredients.

Relationship Types

Entity Name	Multiplicity	Relationship	Multiplicity	Entity Name
Staff	1 .. 1	Receives	0 .. *	Payment
Staff	1 .. 1	Works	1 .. *	Shift
Staff	1 .. 1	Receives	0 .. *	CustomerOrder
Staff	1 .. 1	Places	0 .. *	IngredientOrder
Payment	0 .. 1	For	1 .. 1	Shift
Customer	1 .. 1	Places	1 .. *	CustomerOrder
Customer	1 .. *	Applies	0 .. 1	DiscountProgram
DiscountProgram	0 .. 1	Applies to	1 .. *	CustomerOrder
CustomerOrder	0 .. *	Contains	1 .. *	MenuItem
MenuItem	0 .. *	Consists of	1 .. *	Ingredient
IngredientOrder	0 .. *	Placed for	1 .. *	Ingredient
Supplier	0 .. *	Supplies	0 .. *	Ingredient
Supplier	1 .. 1	Receives	0 .. *	IngredientOrder
DriverStaff	1 .. 1	Makes	0 .. *	Delivery

Attributes

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Staff	EmployeeNo	Uniquely identifies a member of staff.	Integer 4 digits	No	No	No	No
	Name						
	FirstName	Staff member's first name.	15 variable characters	No	No	No	No
	LastName	Staff member's last name.	15 variable characters	No	No	No	No
	PostalAddress						
	StreetNo	Staff member's street number.	10 variable characters	No	No	No	No
	StreetName	Staff member's street name.	20 variable characters	No	No	No	No
	City	Staff member's city or area.	15 variable characters	No	No	No	No
	State	Staff member's state.	20 variable characters	No	No	No	No
	PostCode	Staff member's postal code.	Integer 4 digits	No	No	No	No
	ContactNo	Staff members contact number.	Integer variable 16 digits	No	No	No	No
	TaxFileNo	Staff member's tax file number.	Integer 9 digits	No	No	No	No
	BankDetails						
	AccountNo	Staff member's bank account number.	20 variable characters	No	No	No	No
	BankName	Staff member's bank name.	20 variable characters	No	No	From BSBNo	No
	BSBNo	Staff member's bank BSB number.	Integer 6 digits	No	No	No	No

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Staff (continued)	Status	Identifies if staff member is employed full time or part time.	10 variable characters	No	No	No	No
	Description	Description of the staff member	150 variable characters	Yes	No	No	No
Shift	ShiftNo	Uniquely identifies a shift worked.	Integer 5 digits	No	No	No	No
	ShiftStartDate	The shift start date.	Date type	No	No	No	No
	ShiftEndDate	The shift end date.	Date type	No	No	No	No
	ShiftStartTime	The shift start time.	Time type	No	No	No	No
	ShiftEndTime	The shift end time.	Time type	No	No	No	No
Payment	PaymentID	Uniquely identifies a payment.	Integer 7 digits	No	No	No	No
	Amount	Amount the staff member is paid.	Decimal - 4 digits before decimal place and 2 after	No	No	From PaymentRate and either ShiftStart/ EndTime or OrdersDelivered	No
	DatePaid	The date the payment was made.	Date type	No	No	No	No
Customer	CustomerID	Uniquely identifies a customer.	Integer 4 digits	No	No	No	No
	PhoneNo	Customer's phone number.	Integer variable 16 digits	No	No	No	No
	Name						
	FirstName	Customer's first name.	15 variable characters	No	No	No	No
	LastName	Customer's last name.	15 variable characters	No	No	No	No

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Customer (continued)	PostalAddress						
	StreetNo	Customer's street number.	10 variable characters	No	No	No	No
	StreetName	Customer's street name.	20 variable characters	No	No	No	No
	City	Customer's city or area.	15 variable characters	No	No	No	No
	State	Customer's state.	20 variable characters	No	No	No	No
	PostCode	Customer's postal code.	Integer 4 digits	No	No	No	No
CustomerOrder	OrderNo	Uniquely identifies a customer order.	Integer 5 digits	No	No	No	No
	Date	The date the order was placed.	Date type	No	No	No	No
	DeliveryMethod	Identifies if the order is pick up or delivery.	15 variable characters	No	No	No	Pick up
	PaymentMethod	Identifies the payment method.	15 variable characters	No	No	No	Cash
	OrderTotal	Total that customer pays.	Decimal - 4 digits before decimal place and 2 after	No	No	From SubTotal & DiscountProgram	No
	Tax	Tax that customer pays. (GST)	Decimal - 2 digits before decimal place and 2 after	No	No	From OrderTotal	No
	Status	The delivery status (preparing or delivered)	15 variable characters	No	No	No	Preparing
	PaymentApprovalNo	Order's payment approval number.	Integer 6 digits	Yes	No	No	No

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
CustomerOrder (continue)	DiscountAmount	The amount that is deducted from SubTotal	Decimal - 4 digits before decimal place and 2 after	Yes	No	From DiscountCode & SubTotal	No
	SubTotal	Total amount to pay before Tax and discount.	Decimal - 4 digits before decimal place and 2 after	No	No	From MenuItems	No
DiscountProgram	DiscountCode	Uniquely identifies a discount program.	Integer 6 digits	No	No	No	No
	StartDate	Discount program's start date.	Date type	No	No	No	No
	EndDate	Discount program's end date.	Date type	No	No	No	No
	Requirements	Requirements for discount.	150 variable characters	No	No	No	No
	DiscountPercentage	Discount Percentage.	Integer 2 digits	No	No	No	No
	Description	Discount program description.	150 variable characters	No	No	No	No
MenuItem	ItemNo	Uniquely identifies a menu item.	Integer 4 digits	No	No	No	No
	Name	Name of the menu item.	50 variable characters	No	No	No	No
	Description	Description of the menu item.	150 variable characters	No	No	No	No
	Size	Size of menu item (small, medium or large).	10 variable characters	Yes	No	No	Medium
	CurrentSellingPrice	Menu item's current selling price.	Decimal - 2 digits before decimal place and 2 after	No	No	No	No

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Ingredient	IngredientCode	Uniquely identifies an ingredient.	Integer 4 digits	No	No	No	No
	Name	Name of ingredient.	50 variable characters	No	No	No	No
	Type	Type of ingredient (meat, vegetable etc.)	10 variable characters	No	No	No	No
	Description	Description of ingredient.	150 variable characters	No	No	No	No
	StockLevelAtCurrentPeriod	Stock level at current period	10 variable characters	No	No	No	No
	DateLastStocktakeWasTaken	The date last stocktake was taken	Date type	No	No	No	No
	StockLevelAtLastStocktake	The stock level at last stocktake.	10 variable characters	No	No	No	No
	SuggestedStockLevel	The suggested stock level for ingredient.	10 variable characters	No	No	No	No
IngredientOrder	OrderNo	Uniquely identifies an ingredient order.	Integer 4 digits	No	No	No	No
	DateOrdered	Date the ingredient was ordered.	Date type	No	No	No	No
	DateReceived	Date the order was received by supplier.	Date type	No	No	No	No
	TotalAmount	Total amount to be supplied.	10 variable characters	No	No	No	No
	OrderTotal	Total cost for the ingredient order.	Decimal - 4 digits before decimal place and 2 after	No	No	No	No
	Tax	Tax for the order (GST).	Decimal - 2 digits before decimal place and 2 after	No	No	From OrderTotal	No

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
IngredientOrder (continued)	Status	The status of the order (delivered, processing etc.)	10 variable characters	No	No	No	Processing
	Description	Description of the order.	150 variable characters	No	No	No	No
Supplier	SupplierNo	Uniquely identifies a supplier.	Integer 4 digits	No	No	No	No
	Name	The supplier's name.	50 variable characters	No	No	No	No
	PostalAddress						
	StreetNo	Supplier's street number	10 variable characters	No	No	No	No
	StreetName	Supplier's street name.	20 variable characters	No	No	No	No
	City	Supplier's city or area.	15 variable characters	No	No	No	No
	State	Supplier's state.	20 variable characters	No	No	No	No
	PostCode	Supplier's postal code.	Integer 4 digits	No	No	No	No
	PhoneNo	The supplier's phone number.	Integer variable 16 digits	No	No	No	No
	ContactPerson						
	FirstName	Supplier contact's first name.	15 variable characters	No	No	No	No
	LastName	Supplier contact's last name.	15 variable characters	No	No	No	No
OrderMenuItem (between CustomerOrder & MenuItem)	UnitQuantity	Unit quantity of a menu item in customer order.	Integer 3 digits	Yes	No	No	1
	UnitPrice	Total unit price.	Decimal - 4 digits before decimal place and 2 after	Yes	No	No	No

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
MenuItemIngred-ient (between MenuItem & Ingredient)	IngredientQuantity	Quantity of an ingredient in a menu item.	10 variable characters	No	No	No	No
IngredientsIn-Order	Quantity	The quantity of ingredients in an ingredient order.	15 variable characters	Yes	No	No	No
IngredientSuppl-ier (between Ingredient & Supplier)	SupplierPriority	States if supplier is primary or secondary supplier of an ingredient.	10 variable characters	No	No	No	Secondary
DriverStaff	DriverLicenceNo	Delivery staff member's drivers licence number.	Integer 8 digits	No	No	No	No
	PaymentPerDelivery	Amount driver is paid per delivery in AU dollars.	Decimal – 2 digits before decimal place and 2 after	No	No	No	No
InstoreStaff	PaymentRate	In-store staff member's current payment rate in AU dollars.	Decimal – 2 digits before decimal place and 2 after	No	No	No	No
DriverShift	OrdersDelivered	Amount of orders delivered during a shift.	Integer variable 3 digits	Yes	No	No	No
InstoreShift	HoursWorked	Total hours worked by an in-store employee.	Integer variable 3 digits	Yes	No	No	No
WalkinOrder	CustomerName	Walk in customer's name. Used to verify order.	40 variable characters	No	No	No	No
PhoneOrder	CustomerPhoneNo	Phone number used to place order.	Integer variable 16 digits	No	No	No	No
	OrderVarificationStatus	Indicates if an order has been verified.	15 variable characters	No	No	No	Un-Verified
	VerificationCallStart	Verification call's start time.	Time type	No	No	No	No
	VerificationCallEnd	Verification call's end time.	Time type	No	No	No	No

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
DriverPayment	OrdersDelivered	Amount of orders delivered during a shift.	Integer variable 3 digits	No	No	From DriverShift OrdersDelivered	No
InstorePayment	HoursWorked	Total hours worked by an in-store employee.	Integer variable 3 digits	No	No	From InstoreShift HoursWorked	No
Delivery	DeliveryTime	The time the order was delivered.	Time type	Yes	No	No	No
	DeliveryAddress						
	StreetNo	Delivery street number.	10 variable characters	No	No	No	No
	StreetName	Delivery street name.	20 variable characters	No	No	No	No
	City	Delivery city or area.	15 variable characters	No	No	No	No
	State	Delivery state.	20 variable characters	No	No	No	No
	PostCode	Delivery postal code.	Integer 4 digits	No	No	No	No
Pickup	PickupName	Name of person picking up order.	40 variable characters	No	No	No	No
	PickupTime	Time the order was picked up.	Time type	Yes	No	No	No

Relational Model

```
Staff (EmployeeNo, FirstName, LastName, StreetNo, StreetName, City, State,
      PostCode,   ContactNo, TaxFileNo, AccountNo, BankName, BsbNo,
      Status, Description)
Primary Key       EmployeeNo
Alternate Key     TaxFileNo

InstoreStaff (EmployeeNo, PaymentRate)
Primary Key       EmployeeNo
Foreign Key       EmployeeNo REFERENCES Staff(EmployeeNo) check
                  (EmployeeNo not in DriverStaff) ON UPDATE CASCADE ON
                  DELETE CASCADE

DriverStaff (EmployeeNo, DriversLicenceNo, PaymentPerDelivery)
Primary Key       EmployeeNo
Alternate Key     DriversLicenceNo
Foreign Key       EmployeeNo REFERENCES Staff(EmployeeNo) check
                  (EmployeeNo not in InstoreStaff) ON UPDATE CASCADE ON
                  DELETE CASCADE

Customer (CustomerID, PhoneNo, FirstName, LastName, StreetNo, StreetName,
            City, State, PostCode)
Primary Key       CustomerID

DiscountProgram (DiscountCode, StartDate, EndDate, Requirements,
                  DiscountPercentage, Description)
Primary Key       DiscountCode

CustomerOrder (OrderNo, Date, DeliveryMethod, PaymentMethod,
OrderTotal, DiscountAmount, SubTotal, Tax, Status,
PaymentApprovalNo, DiscountCode)
Primary Key       OrderNo
Alternate Key     PaymentApprovalNo
Foreign Key       DiscountCode REFERENCES DiscountProgram(DiscountCode)
                  check (DiscountProgram (EndDate) >= Date) ON UPDATE
                  CASCADE ON DELETE NO ACTION
```

WalkinOrder (OrderNo, CustomerName)
Primary Key OrderNo
Foreign Key OrderNo **REFERENCES** CustomerOrder(OrderNo) check (OrderNo not in PhoneOrder) **ON UPDATE CASCADE ON DELETE CASCADE**

PhoneOrder (OrderNo, CustomerID, EmployeeNo, CustomerPhoneNo, OrderVerificationStatus, VerificationCallStart, VerificationCallEnd)
Primary Key OrderNo
Foreign Key OrderNo **REFERENCES** CustomerOrder(OrderNo) check (OrderNo not in WalkinOrder) **ON UPDATE CASCADE ON DELETE CASCADE**
Foreign Key CustomerID **REFERENCES** Customer(CustomerID) **ON UPDATE NO ACTION ON DELETE NO ACTION**
Foreign Key EmployeeNo **REFERENCES** Staff(EmployeeNo) check (EmployeeNo not in DriverStaff) **ON UPDATE NO ACTION ON DELETE NO ACTION**

MenuItem (ItemNo, Name, Description, Size, CurrentSellingPrice)
Primary Key ItemNo

OrderMenuItem (OrderNo, ItemNo, UnitQuantity, UnitPrice)
Primary Key (OrderNo, ItemNo)
Foreign Key OrderNo **REFERENCES** CustomerOrder(OrderNo) **ON UPDATE CASCADE ON DELETE CASCADE**
Foreign Key ItemNo **REFERENCES** MenuItem(ItemNo) **ON UPDATE CASCADE ON DELETE CASCADE**

Ingredient (IngredientCode, Name, Type, Description, StockLevelAtCurrentPeriod, DateLastStocktakeWasTaken, StockLevelAtLastStockTake, SuggestedStockLevel)
Primary Key IngredientCode

MenuItemIngredient (ItemNo, IngredientCode, IngredientQuantity)
Primary Key (ItemNo, IngredientCode)
Foreign Key ItemNo **REFERENCES** MenuItem(ItemNo) **ON UPDATE CASCADE ON DELETE CASCADE**
Foreign Key IngredientCode **REFERENCES** Ingredient(IngredientCode) **ON UPDATE CASCADE ON DELETE CASCADE**

Supplier (SupplierNo, Name, StreetNo, StreetName, City, State, PostCode, PhoneNo, FirstName, LastName)
Primary Key SupplierNo

IngredientSupplier (IngredientCode, SupplierNo, SupplierPriority)
Primary Key (IngredientCode, SupplierNo)
Foreign Key IngredientCode **REFERENCES** Ingredient(IngredientCode) **ON UPDATE CASCADE ON DELETE CASCADE**
Foreign Key SupplierNo **REFERENCES** Supplier(SupplierNo) **ON UPDATE CASCADE ON DELETE CASCADE**

IngredientOrder (OrderNo, DateOrdered, DateReceived, TotalAmount, OrderTotal, Tax, Status, Description, SupplierNo)
Primary Key OrderNo
Foreign Key SupplierNo **REFERENCES** Supplier(SupplierNo) **ON UPDATE NO ACTION ON DELETE NO ACTION**

IngredientsInOrder (IngredientCode, OrderNo, quantity)
Primary Key (IngredientCode, OrderNo)
Foreign Key IngredientCode **REFERENCES** Ingredient(IngredientCode) **ON UPDATE CASCADE ON DELETE NO ACTION**
Foreign Key OrderNo **REFERENCES** IngredientOrder(OrderNo) **ON UPDATE CASCADE ON DELETE NO ACTION**

Delivery (OrderNo, EmployeeNo, DeliveryTime, StreetNo, StreetName, City, State, PostCode)
Primary Key OrderNo
Foreign Key OrderNo **REFERENCES** CustomerOrder(OrderNo) check (OrderNo not in Pickup) **ON UPDATE CASCADE ON DELETE NO ACTION**
Foreign Key EmployeeNo **REFERENCES** Staff(EmployeeNo) check (EmployeeNo is in DriverStaff) **ON UPDATE CASCADE ON DELETE NO ACTION**

Pickup (OrderNo, PickupName, PickupTime)
Primary Key OrderNo
Foreign Key OrderNo **REFERENCES** CustomerOrder(OrderNo) check (OrderNo not in Delivery) **ON UPDATE CASCADE ON DELETE NO ACTION**

Shift (ShiftNo, EmployeeNo, ShiftStartDate, ShiftEndDate, ShiftStartTime, ShiftEndTime)
Primary Key ShiftNo
Foreign Key EmployeeNo **REFERENCES** Staff(EmployeeNo) **ON UPDATE CASCADE ON DELETE NO ACTION**

DriverShift (ShiftNo, OrdersDelivered)
Primary Key ShiftNo
Foreign Key ShiftNo **REFERENCES** Shift(ShiftNo) check (ShiftNo not in InstoreShift) **ON UPDATE CASCADE ON DELETE CASCADE**

```
InstoreShift (ShiftNo, HoursWorked)
Primary Key       ShiftNo
Foreign Key       ShiftNo REFERENCES Shift(ShiftNo) check (ShiftNo not in
                  DriverShift) ON UPDATE CASCADE ON DELETE CASCADE

Payment (PaymentID, EmployeeNo, Amount, ShiftNo, DatePaid)
Primary Key       PaymentID
Foreign Key       EmployeeNo REFERENCES Staff(EmployeeNo) check
                  (EmployeeNo == Shift(EmployeeNo)) ON UPDATE CASCADE ON
                  DELETE NO ACTION
Foreign Key       ShiftNo REFERENCES Shift(ShiftNo) ON UPDATE NO ACTION ON
                  DELETE NO ACTION

DriverPayment (PaymentID, OrdersDelivered)
Primary Key       PaymentID
Foreign Key       PaymentID REFERENCES Payment(PaymentID) check (PaymentID
                  not in InstorePayment) ON UPDATE CASCADE ON DELETE
                  CASCADE

InstorePayment (PaymentID, HoursWorked)
Primary Key       PaymentID
Foreign Key       PaymentID REFERENCES Payment(PaymentID) check (PaymentID
                  not in DriverPayment) ON UPDATE CASCADE ON DELETE
                  CASCADE
```

Database Normalisation

Staff

Assume AddressID and BankAccountID exists in Staff and Staff members can have multiple contact numbers.

```
Staff (EmployeeNo, FirstName, LastName, AddressID, StreetNo, StreetName,
      City, State, PostCode,  ContactNo, TaxFileNo, BankAccountID,
      AccountNo, BankName, BsbNo, Status, Description)

FD EmployeeNo →         FirstName, Lastname, AddressID, ContactNo,
                        TaxFileNo, BankAccountID, Status, Description
FD AddressID →          StreetNo, StreetName, City, State, PostCode
FD BankAccountID →      AccountNo, BankName, BsbNo
```

1st Normal Form

The relation is not in 1NF because a staff member could have multiple ContactNo's.

Solution:

```
Staff (EmployeeNo, EmployeeType, FirstName, LastName, AddressID, StreetNo,
      StreetName, City, State, PostCode, TaxFileNo, BankAccountID,
      AccountNo, BankName, BsbNo, Status, Description)
ContactDetails (EmployeeNo, PhoneNo)
```

The relation is now in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form

The Staff table is also already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (EmployeeNo).

3rd Normal Form

The relation is not in 3NF because EmployeeNo -> AddressID and AddressID -> StreetNo, StreetName, City, State and PostCode. Likewise, EmployeeNo -> BankAccountID and BankAccountID -> AccountNo, BsbNo and BankName. Therefore, there are transitive dependencies in Staff.

Solution:

```
Staff (EmployeeNo, EmployeeType, FirstName, LastName, AddressID,
      TaxFileNo, BankAccountID, Status, Description)
StaffAddress (AddressID, StreetNo, StreetName, City, State, PostCode)
BankDetails (BankAccountID, AccountNo, BankName, BsbNo)
```

The relation is now in 3NF since there are no longer any transitive dependencies.

Boyce-Codd Normal Form

The relation is now also in BCNF because every functional dependency is dependent on EmployeeNo (the candidate key).

InstoreStaff

```
InstoreStaff (EmployeeNo, PaymentRate)

FD EmployeeNo → PaymentRate
```

1st Normal Form

The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form

The InstoreStaff table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (EmployeeNo).

3rd Normal Form

The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form

The relation is in BCNF because every functional dependency is dependent on EmployeeNo (the candidate key).

DriverStaff

```
DriverStaff (EmployeeNo, DriversLicenceNo, PaymentPerDelivery)

FD EmployeeNo → DriversLicenceNo, PaymentPerDelivery
```

1st Normal Form

The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The DriverStaff table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (EmployeeNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on EmployeeNo (the candidate key).

Customer
```
Customer (CustomerID, FirstName, LastName, AddressID)

FD CustomerID → FirstName, LastName, AddressID
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The Customer table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (CustomerID).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on CustomerID (the candidate key).

DiscountProgram
```
DiscountProgram (DiscountCode, StartDate, EndDate, Requirements,
DiscountPercentage, Description)

FD DiscountCode →StartDate, EndDate, Requirements, DiscountPercentage,
                  Description
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The DiscountProgram table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (DiscountCode).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on DiscountCode (the candidate key).

CustomerOrder

`CustomerOrder` `(`<u>`OrderNo`</u>`, Date, OrderType, DeliveryMethod, PaymentMethod, OrderTotal, DiscountAmount, SubTotal, Tax, Status, PaymentApprovalNo, DiscountCode)`

```
FD OrderNo →      Date, OrderType, DeliveryMethod, PaymentMethod,
                  OrderTotal, DiscountAmount, SubTotal, Tax, Status,
                  PaymentApprovalNo, DiscountCode
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The CustomerOrder table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (`OrderNo`).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on OrderNo (the candidate key).

WalkinOrder

`WalkinOrder` `(`<u>`OrderNo`</u>`, CustomerName)`

```
FD OrderNo → CustomerName
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The WalkinOrder table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (`OrderNo`).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on OrderNo (the candidate key).

PhoneOrder

`PhoneOrder` `(`<u>`OrderNo`</u>`, CustomerID, EmployeeNo, CustomerPhoneNo, OrderVerificationStatus, VerificationCallStart, VerificationCallEnd)`

```
FD OrderNo →      CustomerID, EmployeeNo, CustomerPhoneNo,
                  OrderVerificationStatus, VerificationCallStart,
                  VerificationCallEnd
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The PhoneOrder table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (`OrderNo`).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on OrderNo (the candidate key).

MenuItem
`MenuItem` `(`<u>`ItemNo`</u>`, Name, Description, Size, CurrentSellingPrice)`

`FD ItemNo → Name, Description, Size, CurrentSellingPrice`

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The MenuItem table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (ItemNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on ItemNo (the candidate key).

OrderMenuItem
`OrderMenuItem` `(`<u>`OrderNo`</u>`, `<u>`ItemNo`</u>`, UnitQuantity, UnitPrice)`

`FD OrderNo, ItemNo → UnitQuantity, UnitPrice`

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The OrderMenuItem table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a composite candidate key ({OrderNo, `ItemNo`}).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on {OrderNo, ItemNo} (the candidate key).

Ingredient
`Ingredient` `(`<u>`IngredientCode`</u>`, Name, Type, Description,`
`StockLevelAtCurrentPeriod, DateLastStocktakeWasTaken,`
`StockLevelAtLastStockTake, SuggestedStockLevel)`

```
FD IngredientCode →     Name, Type, Description,
                        StockLevelAtCurrentPeriod,
                        DateLastStocktakeWasTaken,
                        StockLevelAtLastStockTake, SuggestedStockLevel
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The Ingredient table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (IngredientCode).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on IngredientCode (the candidate key).

MenuItemIngredient
MenuItemIngredient (ItemNo, IngredientCode, IngredientQuantity)

`FD ItemNo, IngredientCode → IngredientQuantity`

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The MenuItemIngredient table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a composite candidate key ({ItemNo, IngredientCode}).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on {ItemNo, IngredientCode} (the candidate key).

Supplier
Supplier (SupplierNo, Name, AddressID, PhoneNo, FirstName, LastName)

`FD SupplierNo → Name, AddressID, PhoneNo, FirstName, LastName`

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The Supplier table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (SupplierNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on SupplierNo (the candidate key).

IngredientSupplier
```
IngredientSupplier (IngredientCode, SupplierNo, SupplierPriority)
```

```
FD IngredientCode, SupplierNo → SupplierPriority
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The IngredientSupplier table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a composite candidate key ({IngredientNo, SupplierNo}).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on {IngredientNo, SupplierNo} (the candidate key).

IngredientOrder
```
IngredientOrder (OrderNo, DateOrdered, DateReceived, TotalAmount,
OrderTotal, Tax, Status, Description, SupplierNo)
```

```
FD OrderNo →      DateOrdered, DateReceived, TotalAmount, OrderTotal, Tax,
                  Status, Description, SupplierNo
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The IngredientOrder table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (OrderNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on OrderNo (the candidate key).

IngredientsInOrder

`IngredientsInOrder (IngredientCode, OrderNo, Quantity)`

`FD IngredientCode, OrderNo → Quantity`

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The IngredientsInOrder table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a composite candidate key ({IngredientCode, OrderNo}).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on {IngredientCode, OrderNo} (the candidate key).

Delivery

`Delivery (OrderNo, EmployeeNo, DeliveryTime, AddressID)`

`FD OrderNo → EmployeeNo, DeliveryTime, AddressID`

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The Delivery table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (OrderNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on OrderNo (the candidate key).

Pickup

`Pickup (OrderNo, PickupName, PickupTime)`

`FD OrderNo → PickupName, PickupTime`

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The Pickup table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (OrderNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on OrderNo (the candidate key).

Shift
```
Shift (ShiftNo, EmployeeNo, ShiftStartDate, ShiftEndDate, ShiftStartTime,
ShiftEndTime)
```

```
FD ShiftNo →     EmployeeNo, ShiftStartDate, ShiftEndDate,
                 ShiftStartTime, ShiftEndTime
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The Shift table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (ShiftNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on ShiftNo (the candidate key).

DriverShift
```
DriverShift (ShiftNo, OrdersDelivered)
```

```
FD ShiftNo → OrdersDelivered
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The DriverShift table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (ShiftNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on ShiftNo (the candidate key).

InstoreShift
```
InstoreShift (ShiftNo, HoursWorked)
```

```
FD ShiftNo → HoursWorked
```

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The InstoreShift table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (ShiftNo).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on ShiftNo (the candidate key).

Payment
Payment (<u>PaymentID</u>, EmployeeNo, Amount, ShiftNo, DatePaid)

FD PaymentID $\rightarrow$ EmployeeNo, Amount, ShiftNo, DatePaid

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The Payment table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (PaymentID).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on PaymentID (the candidate key).

DriverPayment
DriverPayment (<u>PaymentID</u>, OrdersDelivered)

FD PaymentID $\rightarrow$ OrdersDelivered

1st Normal Form
The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form
The DriverPayment table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (PaymentID).

3rd Normal Form
The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form
The relation is in BCNF because every functional dependency is dependent on PaymentID (the candidate key).

InstorePayment

InstorePayment (PaymentID, HoursWorked)

FD PaymentID → HoursWorked

1st Normal Form

The relation is in first normal form because all attributes are single atomic values for their domain.

2nd Normal Form

The InstorePayment table is already in 2NF because all non-candidate key attributes are fully functionally dependent on a candidate key (PaymentID).

3rd Normal Form

The relation is in 3NF since there are no transitive dependencies.

Boyce-Codd Normal Form

The relation is in BCNF because every functional dependency is dependent on PaymentID (the candidate key).

SQL Code

```
--Student No:   c3252194
--Student Name: Jacobus Janse van Vuren

--CREATE DATABASE TABLES

--Create the Address table
CREATE TABLE Address (
    AddressID   INT             PRIMARY KEY CHECK (AddressID > 0 AND AddressID <= 9999),
    StreetNo    VARCHAR(10)     NOT NULL,
    StreetName  VARCHAR(20)     NOT NULL,
    City        VARCHAR(15)     NOT NULL,
    State       VARCHAR(20)     NOT NULL,
    PostCode    INT             NOT NULLCHECK (PostCode > 0 AND PostCode <= 9999)
)

--Create the BankDetails table
CREATE TABLE BankDetails (
    BankAccountID       INT     PRIMARY KEY CHECK (BankAccountID > 0 AND BankAccountID <= 9999),
    AccountNo           VARCHAR(20)     NOT NULL,
    BankName            VARCHAR(20)     NOT NULL,
    BSBNo               INT             NOT NULLCHECK (BSBNo > 0 AND BSBNo <= 999999)
)

--Create the Staff table
CREATE TABLE Staff (
    EmployeeNo    INT           PRIMARY KEY     CHECK (EmployeeNo > 0 AND EmployeeNo <= 9999),
    FirstName     VARCHAR(15)   NOT NULL,
    LastName      VARCHAR(15)   NOT NULL,
    AddressID     INT           NOT NULL CHECK (AddressID > 0 AND AddressID <= 9999),
    ContactNo     INT           NOT NULLCHECK (ContactNo > 0),
    TaxFileNo     INT           NOT NULLCHECK (TaxFileNo > 0 AND TaxFileNo <= 999999999) UNIQUE,
    BankAccountID INT           NOT NULL,       CHECK (BankAccountID > 0 AND BankAccountID <= 9999),
    Status        VARCHAR(10)   NOT NULL,
    Description   VARCHAR(150)
    FOREIGN KEY(AddressID) REFERENCES Address (AddressID) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY(BankAccountID) REFERENCES BankDetails (BankAccountID) ON UPDATE CASCADE ON DELETE
    CASCADE
)
```

```
--Create the InstoreStaff table
CREATE TABLE InstoreStaff (
    EmployeeNo  INT             PRIMARY KEY CHECK (EmployeeNo > 0 AND EmployeeNo <= 9999),
    PaymentRate DECIMAL(4,2)    NOT NULL,
    FOREIGN KEY(EmployeeNo) REFERENCES Staff (EmployeeNo) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the DriverStaff table
CREATE TABLE DriverStaff (
    EmployeeNo          INT     PRIMARY KEY CHECK (EmployeeNo > 0 AND EmployeeNo <= 9999),
    DriverLicenceNo     INT CHECK (DriverLicenceNo > 0 AND DriverLicenceNo <= 99999999) UNIQUE,
    PaymentPerDelivery  DECIMAL(4,2)    NOT NULL,
    FOREIGN KEY(EmployeeNo) REFERENCES Staff (EmployeeNo) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the Customer table
CREATE TABLE Customer (
    CustomerID  INT             PRIMARY KEY CHECK (CustomerID > 0 AND CustomerID <= 9999),
    PhoneNo     INT             NOT NULL CHECK (PhoneNo > 0),
    FirstName   VARCHAR(15)     NOT NULL,
    LastName    VARCHAR(15)     NOT NULL,
    AddressID   INT             NOT NULLCHECK (AddressID > 0 AND AddressID <= 9999),
    FOREIGN KEY(AddressID) REFERENCES Address (AddressID) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the DiscountProgram table
CREATE TABLE DiscountProgram (
    DiscountCode        INT     PRIMARY KEY CHECK (DiscountCode > 0 AND DiscountCode <= 999999),
    StartDate           DATE            NOT NULL,
    EndDate             DATE            NOT NULL,
    Requirements        VARCHAR(150)    NOT NULL,
    DiscountPercentage  DECIMAL(5,4)    NOT NULL CHECK (DiscountPercentage >= 0 AND
    DiscountPercentage <= 1),
    Description         VARCHAR(150)    NOT NULL,
)

--Create the CustomerOrder table
CREATE TABLE CustomerOrder (
    OrderNo             INT     PRIMARY KEY CHECK (OrderNo > 0 AND OrderNo <= 99999),
    Date                DATE            NOT NULL,
    DeliveryMethod      VARCHAR(15)     NOT NULL DEFAULT 'Pick up',
    PaymentMethod       VARCHAR(15)     NOT NULL DEFAULT 'Cash',
    OrderTotal          DECIMAL(6,2)    NOT NULL,
    Tax                 DECIMAL(4,2)    NOT NULL,
    Status              VARCHAR(15)     NOT NULL,
    PaymentApprovalNo   INT             UNIQUE CHECK (PaymentApprovalNo > 0 AND PaymentApprovalNo <=
    999999),
    DiscountAmount      DECIMAL(4,2),
    SubTotal            DECIMAL(6,2)    NOT NULL,
    DiscountCode        INT CHECK (DiscountCode > 0 AND DiscountCode <= 999999),
    FOREIGN KEY(DiscountCode) REFERENCES DiscountProgram(DiscountCode) ON UPDATE CASCADE ON DELETE NO
    ACTION
)

--Create the WalkinOrder table
CREATE TABLE WalkinOrder (
    OrderNo             INT             PRIMARY KEY CHECK (OrderNo > 0 AND OrderNo <= 99999),
    CustomerName        VARCHAR(50)     NOT NULL,
    FOREIGN KEY(OrderNo) REFERENCES CustomerOrder(OrderNo) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the PhoneOrder table
CREATE TABLE PhoneOrder (
    OrderNo                     INT     PRIMARY KEY     CHECK (OrderNo > 0 AND OrderNo <= 99999),
    CustomerID                  INT     NOT NULL CHECK (CustomerID > 0 AND CustomerID <= 9999),
    EmployeeNo                  INT     NOT NULL CHECK (EmployeeNo > 0 AND EmployeeNo <= 9999),
    CustomerPhoneNo             INT     NOT NULL CHECK (CustomerPhoneNo > 0),
    OrderVarificationStatus     VARCHAR(15)     NOT NULLDEFAULT 'Un-varified',
    VerificationCallStart       TIME,
    VerificationCallEnd         TIME,
    FOREIGN KEY(OrderNo) REFERENCES CustomerOrder(OrderNo) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY(CustomerID) REFERENCES Customer(CustomerID) ON UPDATE NO ACTION ON DELETE NO ACTION,
    FOREIGN KEY(EmployeeNo) REFERENCES Staff(EmployeeNo) ON UPDATE NO ACTION ON DELETE NO ACTION
)

--Create the MenuItem table
CREATE TABLE MenuItem (
    ItemNo              INT             PRIMARY KEY CHECK (ItemNo > 0 AND ItemNo <= 9999),
    Name                VARCHAR(50)     NOT NULL,
    Dscription          VARCHAR(150),
    Size                VARCHAR(10)     DEFAULT 'Medium',
    CurrentSellingPrice DECIMAL(4,2)    NOT NULL
)

--Create the OrderMenuItem table
CREATE TABLE OrderMenuItem (
    OrderNo             INT     NOT NULL CHECK (OrderNo > 0 AND OrderNo <= 99999),
    ItemNo              INT     NOT NULL CHECK (ItemNo > 0 AND ItemNo <= 9999),
    UnitQuantity        INT     DEFAULT 1 CHECK (UnitQuantity > 0 AND UnitQuantity <= 999),
    UnitPrice                           DECIMAL(6,2),
    PRIMARY KEY (OrderNo, ItemNo),
    FOREIGN KEY(OrderNo) REFERENCES CustomerOrder(OrderNo) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY(ItemNo) REFERENCES MenuItem(ItemNo) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the Ingredient table
CREATE TABLE Ingredient (
    IngredientCode              INT PRIMARY KEY CHECK (IngredientCode > 0 AND IngredientCode <=
    9999),
    Name                        VARCHAR(50)     NOT NULL,
    Type                        VARCHAR(10)     NOT NULL,
    Description                 VARCHAR(150),
    StockLevelAtCurrentPeriod   VARCHAR(10)     NOT NULL,
    DateLastStocktakeWasTaken   DATE            NOT NULL,
    StockLevelAtLastStocktake   VARCHAR(10)     NOT NULL,
    SuggestedStockLevel         VARCHAR(10)     NOT NULL
)

--Create the MenuItemIngredient table
CREATE TABLE MenuItemIngredient (
    ItemNo              INT             NOT NULL CHECK (ItemNo > 0 AND ItemNo <= 9999),
    IngredientCode      INT             NOT NULL CHECK (IngredientCode > 0 AND IngredientCode <=
    9999),
    IngredientQuantity  VARCHAR(10)     NOT NULL,
    PRIMARY KEY (ItemNo, IngredientCode),
    FOREIGN KEY(ItemNo) REFERENCES MenuItem(ItemNo) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY(IngredientCode) REFERENCES Ingredient(IngredientCode) ON UPDATE CASCADE ON DELETE
    CASCADE
)

--Create the Supplier table
CREATE TABLE Supplier (
    SupplierNo  INT     PRIMARY KEY CHECK (SupplierNo > 0 AND SupplierNo <= 9999),
    Name        VARCHAR(50)             NOT NULL,
    AddressID   INT                     NOT NULLCHECK (AddressID > 0 AND AddressID <= 9999),
    PhoneNo     INT                             NOT NULLCHECK (PhoneNo > 0),
    FirstName   VARCHAR(15)             NOT NULL,
    LastName    VARCHAR(15)             NOT NULL,
    FOREIGN KEY(AddressID) REFERENCES Address (AddressID) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the IngredientSupplier table
CREATE TABLE IngredientSupplier (
    IngredientCode      INT         NOT NULL CHECK (IngredientCode > 0 AND IngredientCode <= 9999),
    SupplierNo          INT         NOT NULL CHECK (SupplierNo > 0 AND SupplierNo <= 9999),
    SupplierPriority    VARCHAR(10) NOT NULL DEFAULT 'Secondary',
    PRIMARY KEY (IngredientCode, SupplierNo),
    FOREIGN KEY(IngredientCode) REFERENCES Ingredient(IngredientCode) ON UPDATE CASCADE ON DELETE
    CASCADE,
    FOREIGN KEY(SupplierNo) REFERENCES Supplier(SupplierNo) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the IngredientOrder table
CREATE TABLE IngredientOrder (
    OrderNo             INT             PRIMARY KEY CHECK (OrderNo > 0 AND OrderNo <= 99999),
    DateOrdered         DATE            NOT NULL,
    DateReceived        DATE            NOT NULL,
    TotalAmount         VARCHAR(10)     NOT NULL,
    OrderTotal          DECIMAL(6,2)    NOT NULL,
    Tax                 DECIMAL(4,2)    NOT NULL,
    Status              VARCHAR(10)     NOT NULL DEFAULT 'Processing',
    Description         VARCHAR(150),
    SupplierNo          INT             NOT NULL CHECK (SupplierNo > 0 AND SupplierNo <= 9999),
    FOREIGN KEY(SupplierNo) REFERENCES Supplier(SupplierNo) ON UPDATE NO ACTION ON DELETE NO ACTION
)

--Create the IngredientsInOrder table
CREATE TABLE IngredientsInOrder (
    IngredientCode      INT     NOT NULL CHECK (IngredientCode > 0 AND IngredientCode <= 9999),
    OrderNo             INT     NOT NULL CHECK (OrderNo > 0 AND OrderNo <= 99999),
    Quantity            VARCHAR(15),
    PRIMARY KEY (IngredientCode, OrderNo),
    FOREIGN KEY(IngredientCode) REFERENCES Ingredient(IngredientCode) ON UPDATE CASCADE ON DELETE NO
    ACTION,
    FOREIGN KEY(OrderNo) REFERENCES IngredientOrder(OrderNo) ON UPDATE CASCADE ON DELETE NO ACTION,
)

--Create the Delivery table
CREATE TABLE Delivery (
    OrderNo             INT     PRIMARY KEY CHECK (OrderNo > 0 AND OrderNo <= 99999),
    EmployeeNo          INT     NOT NULL CHECK (EmployeeNo > 0 AND EmployeeNo <= 9999),
    DeliveryTime        TIME,
    AddressID           INT     NOT NULL CHECK (AddressID > 0 AND AddressID <= 9999),
    FOREIGN KEY(AddressID) REFERENCES Address (AddressID) ON UPDATE NO ACTION ON DELETE NO ACTION,
    FOREIGN KEY(OrderNo) REFERENCES CustomerOrder(OrderNo) ON UPDATE CASCADE ON DELETE NO ACTION,
    FOREIGN KEY(EmployeeNo) REFERENCES DriverStaff(EmployeeNo) ON UPDATE CASCADE ON DELETE NO ACTION
)

--Create the Pickup table
CREATE TABLE Pickup (
    OrderNo     INT             PRIMARY KEY CHECK (OrderNo > 0 AND OrderNo <= 99999),
    PickupName  VARCHAR(40)     NOT NULL,
    PickupTime  TIME,
    FOREIGN KEY(OrderNo) REFERENCES CustomerOrder(OrderNo) ON UPDATE CASCADE ON DELETE NO ACTION
)

--Create the Shift table
CREATE TABLE Shift (
    ShiftNo             INT     PRIMARY KEY CHECK (ShiftNo > 0 AND ShiftNo <= 99999),
    EmployeeNo          INT     NOT NULL CHECK (EmployeeNo > 0 AND EmployeeNo <= 9999),
    ShiftStartDate      DATE,
    ShiftEndDate        DATE,
    ShiftStartTime      TIME,
    ShiftEndTime        TIME,
    FOREIGN KEY(EmployeeNo) REFERENCES Staff(EmployeeNo) ON UPDATE CASCADE ON DELETE NO ACTION
)

--Create the DriverShift table
CREATE TABLE DriverShift (
    ShiftNo             INT     PRIMARY KEY CHECK (ShiftNo > 0 AND ShiftNo <= 99999),
    OrdersDelivered     INT,
    FOREIGN KEY(ShiftNo) REFERENCES Shift(ShiftNo) ON UPDATE CASCADE ON DELETE CASCADE
)
```

```
--Create the InstoreShift table
CREATE TABLE InstoreShift (
    ShiftNo             INT     PRIMARY KEY CHECK (ShiftNo > 0 AND ShiftNo <= 99999),
    HoursWorked         DECIMAL(3,1),
    FOREIGN KEY(ShiftNo) REFERENCES Shift(ShiftNo) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the Payment table
CREATE TABLE Payment (
    PaymentID   INT             PRIMARY KEY CHECK (PaymentID > 0 AND PaymentID <= 9999999),
    EmployeeNo  INT             NOT NULL CHECK (EmployeeNo > 0 AND EmployeeNo <= 9999),
    ShiftNo     INT             NOT NULL CHECK (ShiftNo > 0 AND ShiftNo <= 99999),
    Amount      DECIMAL(6,2)    NOT NULL,
    DatePayed   DATE            NOT NULL,
    FOREIGN KEY(EmployeeNo) REFERENCES Staff(EmployeeNo) ON UPDATE CASCADE ON DELETE NO ACTION,
    FOREIGN KEY(ShiftNo) REFERENCES Shift(ShiftNo) ON UPDATE NO ACTION ON DELETE NO ACTION
)

--Create the DriverPayment table
CREATE TABLE DriverPayment (
    PaymentID           INT     PRIMARY KEY CHECK (PaymentID > 0 AND PaymentID <= 9999999),
    OrdersDelivered     INT,
    FOREIGN KEY(PaymentID) REFERENCES Payment(PaymentID) ON UPDATE CASCADE ON DELETE CASCADE
)

--Create the InstorePayment table
CREATE TABLE InstorePayment (
    PaymentID           INT             PRIMARY KEY CHECK (PaymentID > 0 AND PaymentID <= 9999999),
    HoursWorked         DECIMAL(4,1),
    FOREIGN KEY(PaymentID) REFERENCES Payment(PaymentID) ON UPDATE CASCADE ON DELETE CASCADE
)

GO

--CREATE TRIGGERS

--Check InstoreStaff not in DriverStaff
CREATE TRIGGER check_InstoreStaff
ON InstoreStaff
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @storeEmployeeNo INT
        DECLARE @duplicateCount INT

        SET @storeEmployeeNo = (SELECT EmployeeNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                FROM    DriverStaff d
                                WHERE   d.EmployeeNo = @storeEmployeeNo
                                )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO
```

```
--Check DriverStaff not in InstoreStaff
CREATE TRIGGER check_DriverStaff
ON DriverStaff
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @driverEmployeeNo INT
        DECLARE @duplicateCount INT

        SET @driverEmployeeNo = (SELECT EmployeeNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    InstoreStaff s
                                 WHERE   s.EmployeeNo = @driverEmployeeNo
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO

--Check if Discount code has expired
CREATE TRIGGER check_DiscountCodeExpiry
ON CustomerOrder
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @endDate        DATE
        DECLARE @date                   DATE
        DECLARE @discountCode   INT

        SET @discountCode = (SELECT DiscountCode FROM inserted)
        SET @date = (SELECT Date FROM inserted)
        SET @endDate = (SELECT   EndDate
                         FROM    DiscountProgram dp
                         WHERE   dp.DiscountCode = @discountCode
                         )

        IF @date > @endDate
                ROLLBACK TRANSACTION
END

GO

--Check WalkinOrder not in PhoneOrder
CREATE TRIGGER check_WalkinOrder
ON WalkinOrder
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @walkinOrderNo INT
        DECLARE @duplicateCount INT

        SET @walkinOrderNo = (SELECT OrderNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    PhoneOrder po
                                 WHERE   po.OrderNo = @walkinOrderNo
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO
```

```
--Check PhoneOrder not in WalkinOrder
CREATE TRIGGER check_PhoneOrder
ON PhoneOrder
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @phoneOrderNo INT
        DECLARE @duplicateCount INT

        SET @phoneOrderNo = (SELECT OrderNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    WalkinOrder wo
                                 WHERE   wo.OrderNo = @phoneOrderNo
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO

--Check PhoneOrder EmployeeNo not in DriverStaff
CREATE TRIGGER check_PhoneOrderEmployee
ON PhoneOrder
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @employeeNo INT
        DECLARE @duplicateCount INT

        SET @employeeNo = (SELECT EmployeeNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    DriverStaff ds
                                 WHERE   ds.EmployeeNo = @employeeNo
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO

--Check Delivery EmployeeNo exists in DriverStaff
CREATE TRIGGER check_Delivery
ON Delivery
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @employeeNo INT
        DECLARE @duplicateCount INT

        SET @employeeNo = (SELECT EmployeeNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    DriverStaff ds
                                 WHERE   ds.EmployeeNo = @employeeNo
                                 )

        IF @duplicateCount != 1
                ROLLBACK TRANSACTION
END

GO
```

```sql
--Check Pickup OrderNo not in Delivery
CREATE TRIGGER check_Pickup
ON Pickup
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @orderNo INT
        DECLARE @duplicateCount INT

        SET @orderNo = (SELECT OrderNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    Delivery d
                                 WHERE   d.OrderNo = @orderNo
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO

--Check DriverShift ShiftNo not in InstoreShift
CREATE TRIGGER check_DriverShift
ON DriverShift
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @shiftNo INT
        DECLARE @duplicateCount INT

        SET @shiftNo = (SELECT ShiftNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    InstoreShift i
                                 WHERE   i.ShiftNo = @shiftNo
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO

--Check InstoreShift ShiftNo not in DriverShift
CREATE TRIGGER check_InstoreShift
ON InstoreShift
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @shiftNo INT
        DECLARE @duplicateCount INT

        SET @shiftNo = (SELECT ShiftNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    DriverShift d
                                 WHERE   d.ShiftNo = @shiftNo
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO
```

```
--Check Payment EmployeeNo exists in Shift
CREATE TRIGGER check_Payment
ON Payment
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @employeeNo INT
        DECLARE @duplicateCount INT

        SET @employeeNo = (SELECT EmployeeNo FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    Staff s
                                 WHERE   s.EmployeeNo = @employeeNo
                                 )

        IF @duplicateCount < 1
                ROLLBACK TRANSACTION
END

GO

--Check DriverPayment PaymentID not in InstorePayment
CREATE TRIGGER check_DriverPayment
ON DriverPayment
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @paymentID INT
        DECLARE @duplicateCount INT

        SET @paymentID = (SELECT PaymentID FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    InstorePayment i
                                 WHERE   i.PaymentID = @paymentID
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO

--Check InstorePayment PaymentID not in DriverPayment
CREATE TRIGGER check_InstorePayment
ON InstorePayment
FOR INSERT, UPDATE
AS
BEGIN
        DECLARE @paymentID INT
        DECLARE @duplicateCount INT

        SET @paymentID = (SELECT PaymentID FROM inserted)
        SET @duplicateCount =   (SELECT  COUNT(*)
                                 FROM    DriverPayment d
                                 WHERE   d.PaymentID = @paymentID
                                 )

        IF @duplicateCount > 0
                ROLLBACK TRANSACTION
END

GO
```

```sql
--INSERT DATA INTO TABLES

--Insert Staff address details into Address
INSERT INTO Address VALUES (1, '3', 'Smith Street', 'Newcastle', 'New South Wales', 2300);
INSERT INTO Address VALUES (2, '6', 'Botsford Cutting', 'Newcastle East', 'New South Wales', 2300);
INSERT INTO Address VALUES (3, '322a', 'Bradtke Amble', 'Adamstown', 'New South Wales', 2289);
INSERT INTO Address VALUES (4, '2', 'Catherine Circlet', 'Merewether', 'New South Wales', 2291);
INSERT INTO Address VALUES (5, '13b', 'Braxton Little St.', 'Hillsborough', 'New South Wales', 2290);
INSERT INTO Address VALUES (6, '27', 'Kreiger Ridge', 'Nelson Bay', 'New South Wales', 2315);

--Insert Staff bank account details into BankDetails
INSERT INTO BankDetails VALUES (1, 023454684, 'ANZ', 112298);
INSERT INTO BankDetails VALUES (2, 348374464247, 'Commonwealth Bank', 062903);
INSERT INTO BankDetails VALUES (3, 46813184, 'ING', 923200);
INSERT INTO BankDetails VALUES (4, 46813458, 'NAB', 082976);
INSERT INTO BankDetails VALUES (5, 8768305,'ANZ', 112298);
INSERT INTO BankDetails VALUES (6, 68490756,'ING', 923200);

--Insert details into Staff
INSERT INTO Staff VALUES (1, 'Robert', 'Brown', 1, 0491570156, 865414088, 1, 'Full time', NULL);
INSERT INTO Staff VALUES (2, 'Jeffrey', 'Gottlieb', 2, 0275473375, 459599230, 2, 'Part time' , NULL);
INSERT INTO Staff VALUES (3, 'Freda', 'Conroy', 3, 0262736850, 112474082, 3, 'Full time', NULL);
INSERT INTO Staff VALUES (4, 'Anna', 'Mueller', 4, 0241650502, 565051603, 4, 'Part time', NULL);
INSERT INTO Staff VALUES (5, 'Zakary', 'Shields', 5, 0359696483, 907974668, 5, 'Part time', NULL);
INSERT INTO Staff VALUES (6, 'Granville', 'Greenholt', 6, 0254364468, 907974654, 6, 'Full time',
NULL);

--Insert details into InstoreStaff
INSERT INTO InstoreStaff VALUES (1, 12.5);
INSERT INTO InstoreStaff VALUES (2, 18.0);
INSERT INTO InstoreStaff VALUES (3, 14.0);

--Insert details into DriverStaff
INSERT INTO DriverStaff VALUES (4, 68545980, 4.0);
INSERT INTO DriverStaff VALUES (5, 97356180, 6.0);
INSERT INTO DriverStaff VALUES (6, 77012563, 4.6);

--Insert Customer address details into Address
INSERT INTO Address VALUES (7, '52', 'Kihn Terrace', 'Kotara East', 'New South Wales', 2305);
INSERT INTO Address VALUES (8, '2d', 'Howell Byway', 'Sandgate', 'New South Wales', 2304);
INSERT INTO Address VALUES (9, '4b', 'Jast Estate', 'Cooks Hill', 'New South Wales', 2300);
INSERT INTO Address VALUES (10, '11', 'Prohaska Street', 'Elermore Vale', 'New South Wales', 2287);
INSERT INTO Address VALUES (11, '54', 'Treutel Circle', 'Marks Point', 'New South Wales', 2280);
INSERT INTO Address VALUES (12, '71', 'Jess Slope', 'Wallsend', 'New South Wales', 2287);

--Insert details into Customer
INSERT INTO Customer VALUES (1, 0248518981, 'Guillermo', 'Schumm', 7);
INSERT INTO Customer VALUES (2, 0251341003, 'Lillian', 'Carroll', 8);
INSERT INTO Customer VALUES (3, 0885402466, 'Maya', 'Mosciski', 9);
INSERT INTO Customer VALUES (4, 0880427468, 'Ian', 'Denesik', 10);
INSERT INTO Customer VALUES (5, 0255777323, 'Jalen', 'Lowe', 11);
INSERT INTO Customer VALUES (6, 0243918997, 'Ramona', 'Blick', 12);

--Insert details into DiscountProgram
INSERT INTO DiscountProgram VALUES (1, '2017-08-01', '2017-10-21', 'Buy a cheese pizza', 0.25, 'Buy a
cheese pizza and get 25% off');
INSERT INTO DiscountProgram VALUES (2, '2017-10-01', '2017-11-01', 'Buy two pizzas', 0.10, 'Buy two
pizzas and get 10% off');
INSERT INTO DiscountProgram VALUES (3, '2017-10-16', '2017-10-27', 'Buy a hawaiian pizza', 0.30, 'Buy
a hawaiian pizza and get 30% off');

--Insert details into CustomerOrder
INSERT INTO CustomerOrder VALUES (1, '2017-10-20', 'Pick up', 'Cash', 22.55, 2.05, 'Delivered', 1,
0.00, 20.50, NULL);
INSERT INTO CustomerOrder VALUES (2, '2017-10-20', 'Pick up', 'Savings', 35.20, 3.20, 'Delivered', 2,
0.00, 32.00, NULL);
INSERT INTO CustomerOrder VALUES (3, '2017-10-20', 'Pick up', 'Savings', 9.35, 0.85, 'Delivered', 3,
2.50, 10.00, 1);
```

```
INSERT INTO CustomerOrder VALUES (4, '2017-10-20', 'Pick up', 'Cash', 17.05, 1.55, 'Delivered', 4,
0.00, 15.50, NULL);
INSERT INTO CustomerOrder VALUES (5, '2017-10-20', 'Pick up', 'Credit', 27.50, 2.50, 'Delivered', 5,
0.00, 25.00, NULL);
INSERT INTO CustomerOrder VALUES (6, '2017-10-20', 'Pick up', 'Credit', 35.15, 3.20, 'Delivered', 6,
3.55, 35.50, 2);
INSERT INTO CustomerOrder VALUES (7, '2017-10-20', 'Delivery', 'Credit', 13.20, 1.20, 'Delivered', 7,
0.00, 12.00, NULL);
INSERT INTO CustomerOrder VALUES (8, '2017-10-20', 'Delivery', 'Credit', 20.08, 1.83, 'Delivered', 8,
0.00, 18.25, NULL);
INSERT INTO CustomerOrder VALUES (9, '2017-10-20', 'Delivery', 'Credit', 17.40, 1.58, 'Delivered', 9,
6.78, 22.60, 3);

--Insert details into WalkinOrder
INSERT INTO WalkinOrder VALUES (1, 'Jo');
INSERT INTO WalkinOrder VALUES (2, 'Mo');
INSERT INTO WalkinOrder VALUES (3, 'Bo');

--Insert details into PhoneOrder
INSERT INTO PhoneOrder VALUES (4, 1, 1, 0248518981, 'Verified', '17:40', '17:42');
INSERT INTO PhoneOrder VALUES (5, 2, 2, 0251341003, 'Verified', '17:55', '17:56');
INSERT INTO PhoneOrder VALUES (6, 3, 2, 0885402466, 'Verified', '18:00', '18:02');
INSERT INTO PhoneOrder VALUES (7, 4, 1, 0880427468, 'Verified', '18:23', '18:25');
INSERT INTO PhoneOrder VALUES (8, 5, 1, 0255777323, 'Verified', '18:48', '18:49');
INSERT INTO PhoneOrder VALUES (9, 6, 1, 0243918997, 'Verified', '19:30', '19:31');

--Insert details into MenuItem
INSERT INTO MenuItem VALUES (1, 'Cheese Pizza', 'A plain cheese pizza', 'Medium', 5.00);
INSERT INTO MenuItem VALUES (2, 'Hawaiian Pizza', 'Pizza containing pinapple, ham and cheese',
'Medium', 8.95);
INSERT INTO MenuItem VALUES (3, 'Pepperoni Pizza', 'Pizza containing pepperoni and cheese', 'Medium',
5.00);

--Insert details into OrderMenuItem
INSERT INTO OrderMenuItem VALUES (3, 1, 2, 10.0);
INSERT INTO OrderMenuItem VALUES (5, 3, 3, 15.0);
INSERT INTO OrderMenuItem VALUES (2, 2, 2, 37.9);

--Insert details into Ingredient
INSERT INTO Ingredient VALUES (1, 'Pizza Dough', 'Dough', 'Used to make the pizza crust', '156 kg',
'2017-10-15', '25 kg', '300 kg');
INSERT INTO Ingredient VALUES (2, 'Tomato Sauce', 'Sauce', 'Base Pizza sauce', '200 l', '2017-10-15',
'75 l', '300 l');
INSERT INTO Ingredient VALUES (3, 'Mozzarella Cheese', 'Dairy', 'Southern Italian dairy product made
from Italian buffalos milk', '100 kg', '2017-10-15', '50 kg', '150 kg');
INSERT INTO Ingredient VALUES (4, 'Ham', 'Meat', 'Ham is pork that has been preserved through salting,
smoking, or wet curing.', '50 kg', '2017-10-15', '5 kg', '120 kg');
INSERT INTO Ingredient VALUES (5, 'Pinapple', 'Fruit', 'A tropical fruit', '40 kg', '2017-10-15', '12
kg', '90 kg');
INSERT INTO Ingredient VALUES (6, 'Pepperoni', 'Meat', 'An American variety of salami', '20 kg',
'2017-10-15', '61 kg', '120 kg');

--Insert details into MenuItemIngredient
INSERT INTO MenuItemIngredient VALUES (1, 1, '460 g');
INSERT INTO MenuItemIngredient VALUES (1, 2, '141 g');
INSERT INTO MenuItemIngredient VALUES (1, 3, '227 g');
INSERT INTO MenuItemIngredient VALUES (2, 1, '460 g');
INSERT INTO MenuItemIngredient VALUES (2, 2, '141 g');
INSERT INTO MenuItemIngredient VALUES (2, 3, '227 g');
INSERT INTO MenuItemIngredient VALUES (2, 4, '75 g');
INSERT INTO MenuItemIngredient VALUES (2, 5, '82 g');
INSERT INTO MenuItemIngredient VALUES (3, 1, '460 g');
INSERT INTO MenuItemIngredient VALUES (3, 2, '141 g');
INSERT INTO MenuItemIngredient VALUES (3, 3, '227 g');
INSERT INTO MenuItemIngredient VALUES (3, 6, '85 g');

--Insert Supplier address details into Address
INSERT INTO Address VALUES (13, '112', 'Kieran Street', 'Wallsend', 'New South Wales', 2287);
INSERT INTO Address VALUES (14, '14', 'Kemmer Street', 'Charlestown', 'New South Wales', 2290);
INSERT INTO Address VALUES (15, '86', 'Volkman Alley', 'Kotara', 'New South Wales', 2289);

--Insert details into Supplier
INSERT INTO Supplier VALUES (1, 'Joes Premium Meats', 13, 0277392976, 'Joe', 'Cormier');
INSERT INTO Supplier VALUES (2, 'Newcastle Dairy', 14, 0218040164, 'Colin', 'Crona');
INSERT INTO Supplier VALUES (3, 'Heldas Bakery', 15, 0290268071, 'Helda', 'Towne');

--Insert details into IngredientSupplier
INSERT INTO IngredientSupplier VALUES (1, 3, 'Primary');
INSERT INTO IngredientSupplier VALUES (3, 2, 'Primary');
INSERT INTO IngredientSupplier VALUES (4, 1, 'Primary');
INSERT INTO IngredientSupplier VALUES (6, 1, 'Primary');

--Insert details into IngredientOrder
INSERT INTO IngredientOrder VALUES (1, '2017-10-15', '2017-10-17', '50 kg', 80.0, 8.0, 'Delivered',
'Order for mozzarella cheese', 2);
INSERT INTO IngredientOrder VALUES (2, '2017-10-20', '2017-10-22', '100 kg', 250.0, 25.0, 'Delivered',
'Order for pepperoni and ham', 1);
INSERT INTO IngredientOrder VALUES (3, '2017-10-22', '2017-10-23', '30 kg', 50.0, 5.0, 'Delivered',
'Order for pizza dough', 3);

--Insert details into IngredientsInOrder
INSERT INTO IngredientsInOrder VALUES (3, 1, '50 kg');
INSERT INTO IngredientsInOrder VALUES (4, 2, '50 kg');
INSERT INTO IngredientsInOrder VALUES (6, 2, '50 kg');
INSERT INTO IngredientsInOrder VALUES (1, 3, '30 kg');

--Insert Delivery address details into Address
INSERT INTO Address VALUES (16, '14b', 'Ressie Street', 'Hillsborough', 'New South Wales', 2290);
INSERT INTO Address VALUES (17, '84', 'Jacobs Avenue', 'Black Hill', 'New South Wales', 2322);
INSERT INTO Address VALUES (18, '56', 'Lazaro Crossroad', 'Bar Beach', 'New South Wales', 2300);

--Insert details into Delivery
INSERT INTO Delivery VALUES (7, 4, '18:00', 16);
INSERT INTO Delivery VALUES (8, 5, '18:25', 17);
INSERT INTO Delivery VALUES (9, 6, '19:51', 18);

--Insert details into Pickup
INSERT INTO Pickup VALUES (1, 'Jo', '17:30');
INSERT INTO Pickup VALUES (2, 'Mo', '18:21');
INSERT INTO Pickup VALUES (3, 'Bo', '19:43');

--Insert details into Shift
INSERT INTO Shift VALUES (1, 1, '2017-10-21', '2017-10-21', '09:00', '17:00');
INSERT INTO Shift VALUES (2, 2, '2017-10-21', '2017-10-21', '16:00', '20:00');
INSERT INTO Shift VALUES (3, 3, '2017-10-21', '2017-10-21', '09:00', '17:00');
INSERT INTO Shift VALUES (4, 4, '2017-10-22', '2017-10-22', '16:00', '21:00');
INSERT INTO Shift VALUES (5, 5, '2017-10-22', '2017-10-22', '17:30', '21:30');
INSERT INTO Shift VALUES (6, 6, '2017-10-22', '2017-10-22', '09:00', '17:00');

--Insert details into DriverShift
INSERT INTO DriverShift VALUES (4, 8);
INSERT INTO DriverShift VALUES (5, 4);
INSERT INTO DriverShift VALUES (6, 20);

--Insert details into InstoreShift
INSERT INTO InstoreShift VALUES (1, 8.0);
INSERT INTO InstoreShift VALUES (2, 4.0);
INSERT INTO InstoreShift VALUES (3, 8.0);

--Insert details into Payment
INSERT INTO Payment VALUES (1, 1, 1, 100.0, '2017-10-22');
INSERT INTO Payment VALUES (2, 2, 2, 72.0, '2017-10-22');
INSERT INTO Payment VALUES (3, 3, 3, 112.0, '2017-10-22');
INSERT INTO Payment VALUES (4, 4, 4, 32.0, '2017-10-23');
INSERT INTO Payment VALUES (5, 5, 5,  24.0,'2017-10-23');
INSERT INTO Payment VALUES (6, 6, 6, 92.0, '2017-10-23');
```

```
--Insert details into DriverPayment
INSERT INTO DriverPayment VALUES (4, 8);
INSERT INTO DriverPayment VALUES (5, 4);
INSERT INTO DriverPayment VALUES (6, 20);

--Insert details into InstorePayment
INSERT INTO InstorePayment VALUES (1, 8.0);
INSERT INTO InstorePayment VALUES (2, 4.0);
INSERT INTO InstorePayment VALUES (3, 8.0);

GO

--QUERIES

--Query 1
--For  a  staff  with  id  number  xxx,  print  his/her  1stname, lname, and hourly payment rate.
SELECT  FirstName, LastName, PaymentRate
FROM    Staff s INNER JOIN InstoreStaff i ON s.EmployeeNo = i.EmployeeNo
WHERE   i.EmployeeNo = 1

--Query 2
--List the ingredient details of a menu item named xxx.
SELECT  i.*
FROM    MenuItem m       INNER JOIN MenuItemIngredient mi ON m.ItemNo = mi.ItemNo
                         INNER JOIN Ingredient i ON mi.IngredientCode = i.IngredientCode
WHERE   m.Name = 'Cheese Pizza'

--Query 3
--List all the order details of the orders that are made by the customer with first name xxx
--via phone between date yyy and zzz.
SELECT  co.OrderNo, Date, DeliveryMethod, PaymentMethod, OrderTotal, Tax, Status, PaymentApprovalNo,
        DiscountAmount, SubTotal, DiscountCode
FROM    Customer c       INNER JOIN PhoneOrder p ON c.CustomerID = p.CustomerID
                         INNER JOIN CustomerOrder co ON p.OrderNo = co.OrderNo
WHERE   c.FirstName = 'Guillermo' AND co.Date > '2017-10-15' AND co.Date < '2017-10-25'

--Query 4
--Print  the  salary  paid  to  a  delivery  staff  named  xxx  in current month.
SELECT  SUM(Amount) AS 'Salary Paid'
FROM    Staff s INNER JOIN DriverStaff ds ON s.EmployeeNo = ds.EmployeeNo
                INNER JOIN Payment p ON ds.EmployeeNo = p.EmployeeNo
WHERE   FirstName = 'Anna' AND LastName = 'Mueller' AND MONTH(DatePayed) = MONTH(getdate())

--Query 5
--List the menu item that is mostly ordered in current month.
SELECT  m.ItemNo, m.Name
FROM    CustomerOrder co INNER JOIN OrderMenuItem om ON co.OrderNo = om.OrderNo
                         RIGHT JOIN MenuItem m ON om.ItemNo = m.ItemNo
WHERE   UnitQuantity  >= ALL     (SELECT SUM(UnitQuantity)
                                 FROM    CustomerOrder co INNER JOIN OrderMenuItem om ON co.OrderNo =
                                         om.OrderNo
                                                          RIGHT JOIN MenuItem m ON om.ItemNo = m.ItemNo
                                 WHERE   MONTH(co.Date) = MONTH(getdate())
                                 GROUP BY m.ItemNo
                                 )
GROUP BY m.ItemNo, m.Name

--Query 6
--List   the   ingredient(s)   that   was/were   supplied   by   the supplier with supplier ID xxx on
date yyy
SELECT  i.IngredientCode, i.Name
FROM    IngredientsInOrder ii INNER JOIN Ingredient i ON ii.IngredientCode = i.IngredientCode
WHERE   ii.OrderNo =     (SELECT OrderNo
                          FROM   IngredientOrder io INNER JOIN Supplier s ON io.SupplierNo =
                                 s.SupplierNo
                          WHERE  s.SupplierNo = 2 AND DateReceived = '2017-10-17'
                         )
```