Assignment 3
Mega Pizza Project – Logical Database Design

Table of Contents

Requirements Specification

Data Requirements

Order Processing

- **Order:** This entity/data is used to store the details on an Order which includes attributes OrderNo, OrderDateTime, OrderStatus (the status of the order), Total Amount Due, Payment Method, paymentApprovalNo, Description. There are two types of order that can be placed these include:
  - A **WalkInOrder** in which a customer has placed an order in person at the store. This data stores the time that the customer entered the store to order (TimeWalkedIn).
  - A **PhoneOrder** is when a customer has decided to place an order via phone. The data stored in this field is TimeAnsweredCall, TimeTerminatedCall. The PhoneOrder can be further spilt into two types which include:
    - **PickupOrder** in which the customer has called the store to place an order and has selected to pick up the order at the store. The attributes for this data include the pickUpTime (the time the order was picked up from the store)
    - **DeliveryOrder** in which the customer has selected to have the pizza's delivered to them. The attributes for this entity include deliveryTime, deliveryAddress and DriverStaffInfo.
- **Customer:** A customer is data/entity that is used to store the following details about the customer. CusId, fName, lName, placeOfResident, Phone, Status.

Menu Items, Ingredients and Suppliers

- **MenuItems:** The store maintains details about each menu item that is available for order. The details are made up of ItemCode, Name, PizzaSize, CurrentSellingPrice and Description.

- **Ingredients:** This ingredient data stores the details about an ingredient that is used for making certain pizzas. The details includes Code, Name, IngredientType, Description, StockLevelAtCurrentPeriod, StockLevelAtLastStocktake, SuggestedStockLevel, ReorderLevel, DateLastStocktakeTaken, shelfLife.

- **Supplier:** An ingredient can be supplied by a supplier so this data/entity is used to store the details about a supplier. The attributes in this field include SupplierNo, Name, Address, Phone, Email, and Contact Person.

- **IngredientOrder:** When an ingredient stock level is below the suggested stock level then an Ingredient Order is created to get the ingredient. A supplier provides the ingredient for an Ingredient Order. This attributes included in this entity is the information about the ingredient order which includes OrderNo, DateOfOrder, DateRecievedOrder, TotalAmount, OrderStatus, Description.

- **Staff:** This data stores the details about a staff member that is working for the pizza outlet and is made up of EmployeeNumber, Firstname, Lastname, PostalAddress, ContactAddress, TaxFileNumber, BankDetails, PaymentRate, workStatus, and Description. Bank details is a composite attribute which consists of bankCode, bankName, bankAccountNumber. The store has two types of staff members:
  - **ShopStaff:** This data is used to specific details about a shop staff member. This data includes the role of the staff member in the store (shopRole). The role can be many things such as cashier, pizza maker, manager, trainer.. etc.
  - **DriverStaff:** This data is used to store the details of a delivery staff member. The LicenceNumber of a delivery staff in stored in this field.

- **Shift:** A staff member completes shifts for the store so this data will store the details of a shift that was done by a staff member. The details stored are shiftId, start date, start time, end date, end time. There are two types of shift:
  - **ShopStaffShift**: This is used to store the details about a shop staff member's shift. The attributes for this data are breakTime. The reason behind the break time is that a shop staff member will not be working from start to end time (say from 8am to 8pm) as they require break time during the shift.
  - **DeliveryStaffShift**: This is used to store the details that about the delivery staff members shift. This includes information such as the amount of deliveries (NumOfDeliveries) that are made by delivery staff member during the shift.

- **PaymentRecord:** This entity is used to store the payment record of the employee working at the store. The attributes of this entity are Payment record ID, Gross payment, Tax Withheld, Total Amount Paid, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate. There are two types of payment that are made to a staff member:
  - **DriverStaffPayment:** This entity is used to payment record of a driver staff member. Although there isn't anything specified about this type of record. We can assume that TotalNumberOfDeliveries is kept in the entity
  - **StoreStaffPayment:** This entity is used to payment record of a driver staff member. Although there isn't anything specified about this type of record. We can assume that TotalNumberOfHoursWorked is kept in the entity.

<u>Data Manipulation</u>

<u>Data Entry:</u>

- Insert an Order
- Insert the results of the weekly stocktake
- Insert an Ingredient Order into the database. This can only be done by the store manager.
- Insert a customer into the database.
- Insert a shift for a staff member

Data update/deletion:

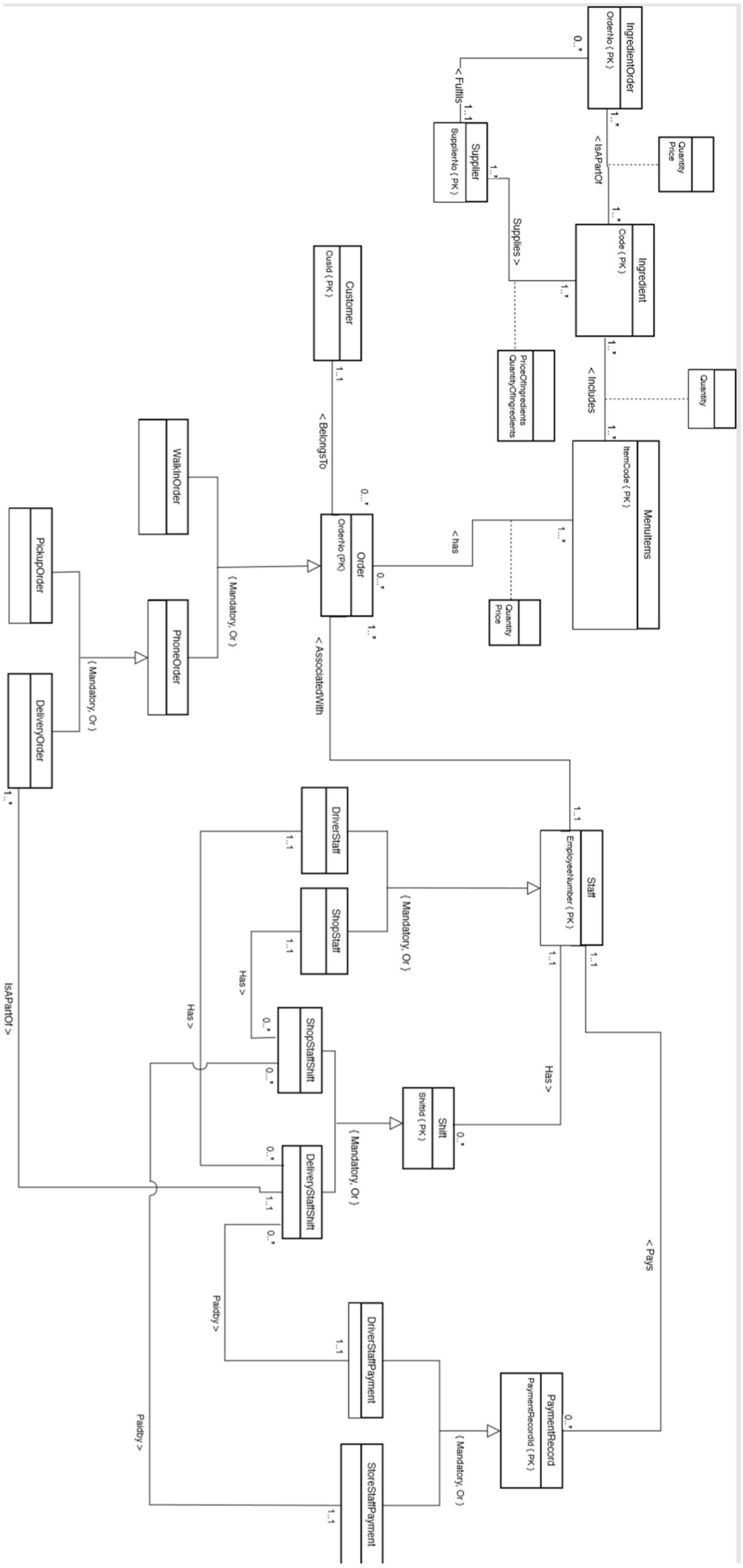
- Update/Delete an existing Order.
- Update StockLevelAtCurrentPeriod for an ingredient
- Update the status of a customer, this is done if the customer is new and has not been verified yet
- Update an order status of an order entity, could be on hold if the customer has not been verified yet.
- Update the paymentApprovalNo of an order. This is done when the customer decides to pay using card.
- Update the number of deliveries made by a delivery staff member in a shift.
- Update the bank details of an employee. A staff member may decide to change their bank details during their employment at the company.

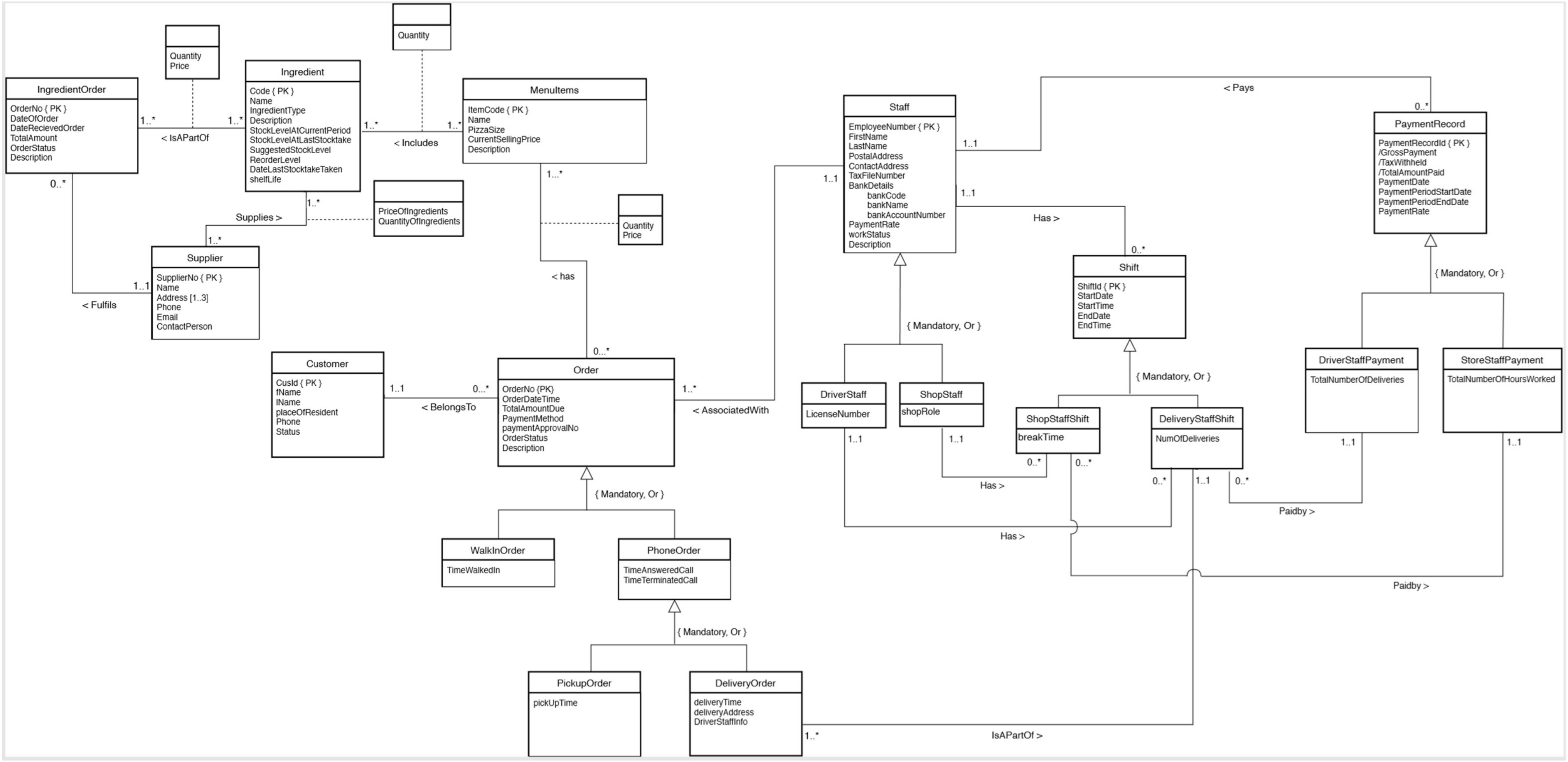
Data queries:

- Search a payment based on an employee number, on a particular pay date
- Search a delivery based in a customer number, on a particular date.
- List ingredients and quantity used for menu items
- Report of ingredient levels for current period and suggested stock levels
- Search a customer’s name and address based on the customer phone number.
- Get a customer’s information from an order based on the order ID. This is used to verify whether the customer is hoax or not.

- The amount of each ingredient remaining must be updated every time some of the ingredient 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.
- 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 or 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 order's type can only be one of the following:
  - Pick up
  - Delivery
  - Walk In
  - Phone Order
- If an order is paid for using a card, the approval number must be stored in the order's paymentApprovalNo.
- If the customer name and address do not correspond with what is recorded in the database then a new customer is created.
- A supplier can provide more than one type of ingredient to the store.
- An ingredient can have more than one supplier.
- An Ingredient Order can only be placed by a store manager.
- Only one driver can be associated with a delivery order.
- A delivery driver can have multiple delivery orders.
- Only one staff member can be associated with an order for a customer.
- Each ingredient can be provided by multiple suppliers.
- If a customer places a Walk in Order then the order can only be picked up from the store.
- An order can only belong to one customer.
- When an order is created, the default for paymentApprovalNo is set to NULL. It is changed if the customer decides to pay by card.

Part 2: EER Model with Data Dictionary




IngredientOrder
OrderNo { PK }
DateOfOrder
DateRecievedOrder
TotalAmount
OrderStatus
Description
Quantity
Price
< IsAPartOf
1..*
1..*
0..*
Ingredient
Code { PK }
Name
IngredientType
Description
StockLevelAtCurrentPeriod
StockLevelAtLastStocktake
SuggestedStockLevel
ReorderLevel
DateLastStocktakeTaken
shelfLife
Quantity
< Includes
1..*
1..*
MenuItems
ItemCode { PK }
Name
PizzaSize
CurrentSellingPrice
Description
1...*
Supplies >
1..*
1..*
PriceOfIngredients
QuantityOfIngredients
Quantity
Price
< has
Supplier
SupplierNo { PK }
Name
Address [1..3]
Phone
Email
ContactPerson
1..1
< Fulfils
Customer
CusId { PK }
fName
lName
placeOfResident
Phone
Status
1..1
< BelongsTo
0...*
0...*
Order
OrderNo {PK}
OrderDateTime
TotalAmountDue
PaymentMethod
paymentApprovalNo
OrderStatus
Description
1..*
< AssociatedWith
{ Mandatory, Or }
WalkInOrder
TimeWalkedIn
PhoneOrder
TimeAnsweredCall
TimeTerminatedCall
{ Mandatory, Or }
PickupOrder
pickUpTime
DeliveryOrder
deliveryTime
deliveryAddress
DriverStaffInfo
1..*
IsAPartOf >
Staff
EmployeeNumber { PK }
FirstName
LastName
PostalAddress
ContactAddress
TaxFileNumber
BankDetails
bankCode
bankName
bankAccountNumber
PaymentRate
workStatus
Description
1..1
1..1
1..1
< Pays
0..*
Has >
0..*
PaymentRecord
PaymentRecordId { PK }
/GrossPayment
/TaxWithheld
/TotalAmountPaid
PaymentDate
PaymentPeriodStartDate
PaymentPeriodEndDate
PaymentRate
{ Mandatory, Or }
Shift
ShiftId { PK }
StartDate
StartTime
EndDate
EndTime
{ Mandatory, Or }
{ Mandatory, Or }
DriverStaff
LicenseNumber
ShopStaff
shopRole
1..1
1..1
ShopStaffShift
breakTime
0..*
0...*
DeliveryStaffShift
NumOfDeliveries
0..*
1..1
0..*
Has >
Has >
DriverStaffPayment
TotalNumberOfDeliveries
StoreStaffPayment
TotalNumberOfHoursWorked
1..1
1..1
Paidby >
Paidby >

Data Dictionary

Entities

Entity Name	Description	Aliases	Occurrence
Order	Describing the order that customers have made	Pizza Order	When an order is made by a customer
WalkInOrder	Describing the details of an in-store order	In-Store Order	When a customer places an order in person at the store
PhoneOrder	Describing the details of an order placed via phone by a customer	Phone Order	When a customer places an order via phone
PickupOrder	Describing the details of a pickup order placed via phone by a customer	Pickup Phone Order	When a customer places an order via phone and decided to pick it up at the store
DeliveryOrder	Describing the details of a delivery order placed via phone by a customer	Delivery Phone Order	When a customer places an order via phone and decided to have it delivered
Customer	Describing the details of a customer	Client	A new or previous customer of the store is ordering pizzas from the store
MenuItems	Describing the pizzas that are available for order	Menu Item	A set of the pizza's available for order
Ingredient	Describing the details about an ingredient used by the store to make pizzas.	Component	A component of certain pizza's made by the store
Supplier	Describing the details of a supplier	Provider	A company or person that supplies ingredients to the store

Entity Name	Description	Aliases	Occurrence
IngredientOrder	Describing the details of an ingredient order	Stock Order	When the current stock level of an ingredient is lower than the suggested stock level
Staff	Describing the details of a staff member that works for the outlet	Employee	An individual/person that works for the store
ShopStaff	Describing the details specific to a Store Staff Member	In-Store Employee	A staff member is a in-store member for the store
DriverStaff	Describing the details specific to a Driver Staff Member	Delivery Employee, Driver	A staff member is a driver for the store
Shift	Describing details of a shift performed by an employee	Working Time	When a staff member performs a shift
ShopStaffShift	Describes the details of a shift performed by a Shop Staff Member	In-Store Shift	When a Shop Staff Member performs a shift
DeliveryStaffShift	Describes the details of a shift performed by a Delivery Staff Member	Delivery Shift	When a Delivery Staff Member performs a shift
PaymentRecord	Describes the details of a employees payment record	Payment Record	When a new employee starts working for the store
DriverStaffPayment	Describes the details of a Delivery Staff Member's payment record	Delivery Staff Record	When a new driver staff member commences work for the store
StoreStaffPayment	Describes the details of a Store Staff Member's payment record	Store Staff Record	When a new shop staff member commences work for the store

Relationships

Entity Name	Multiplicity	Relationship	Multiplicity	Entity Name
Order	0..*	Has	1..*	MenuItems
	(Mandatory, Or)	Generalisation	(Mandatory, Or)	WalkInOrder
	(Man, Or)	Generalisation	(Mandatory, Or)	PhoneOrder
	0..*	BelongsTo	1..1	Customer
PhoneOrder	(Mandatory, Or)	Generalisation	(Mandatory, Or)	DeliveryOrder
	(Mandatory, Or)	Generalisation	(Mandatory, Or)	PickUpOrder
DeliveryOrder	1..*	IsAPartOf	1..1	DeliveryStaffShift
MenuItems	1..*	Includes	1..*	Ingredient
Ingredient	1..*	IsAPartOf	1..*	IngredientOrder
Supplier	1..*	Supplies	1..*	Ingredient
	1..1	Fulfils	0..*	IngredientOrder
Staff	1..1	AssociatedWith	1..*	Order
	(Mandatory, Or)	Generalisation	(Mandatory, Or)	ShopStaff
	(Mandatory, Or)	Generalisation	(Mandatory, Or)	DriverStaff
	1..1	Has	0..*	Shift
Shift	(Mandatory, Or)	Generalisation	(Mandatory, Or)	DeliveryStaffShift
	(Mandatory, Or)	Generalisation	(Mandatory, Or)	ShopStaffShift
PaymentRecord	0..*	Pays	1..1	Staff
	(Mandatory, Or)	Generalisation	(Mandatory, Or)	DriverStaffPayment
	(Mandatory, Or)	Generalisation	(Mandatory, Or)	StoreStaffPayment

Entity Name	Multiplicity	Relationship	Multiplicity	Entity Name
DriverStaff	1..1	Has	0..*	DeliveryStaffShift
ShopStaff	1..1	Has	0..*	ShopStaffShift
ShopStaffShift	0..*	PaidBy	1..1	StoreStaffPayment
DeliveryStaffShift	0..*	PaidBy	1..1	DriverStaffPayment

Descriptive Attribute Relationships

Entity Name	Multiplicity	Relationship	Multiplicity	Entity Name
QSupplierIngredient		Relationship Variables		(Ingredient side 1…*, Supplier side 1…*)
QOrderMenuItems		Relationship Variables		(MenuItems side 1…*, Order side 1…*)
QMenuItemsIngredient		Relationship Variables		(MenuItems side 1…*, Ingredient side 1…*)
QIngredient&IngredientOrder		Relationship Variables		(Ingredient side 1…*, IngredientOrder side 1…*)

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
QSupplierIngredient	PriceOfIngrents	The price that the supplier offers for an ingredient	Float	Not Null	N	N	
	QuantityOfIngredients	The amount of ingredients being supplied	Integer	Not Null	N	N	1
QOrderMenuItems	Quantity	The quantity of menu item in the order	Integer	Not Null	N	N	1
	Price	The price of each menu Item	Float	Not Null	N	N	
QMenuItemsIngredient	Quantity	The quantity of ingredient in the menu item	Integer	Not Null	N	N	1
QIngredient&IngredientOrder	Quantity	The quantity of each ingredient in the Ingredient Order	Integer	Not Null	N	N	1
	Price	The price of each ingredient in the Ingredient Order	Float	Not Null	N	N	

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Order	OrderNo	Unique Order identifier	Char(10)	N	N	N	
	OrderDateTime	The date and time the order is made	datetime	Y	N	N	
	TotalAmountDue	The total amount due for the order	Float	Y	N	N	
	PaymentMethod	The method of payment	Varchar(20)	Y	N	N	
	paymentApprovalNo	A payment approval number is taken if the payment of the order is done via card	Char(10)	Y	N	N	Null
	OrderStatus	The status of an order	Varchar(10)	Y	N	N	
	Description	description of an Order	Char(30)	Y	N	N	

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Customer	CusId	Unique Order identifier	Char(10)	N	N	N	
	fName	First name of the customer	Char(15)	Y	N	N	
	lName	Last name of the customer	Char(15)	Y	N	N	
	placeOfResident	Address	Char(50)	Y	N	N	
	Phone	Phone Number	INTEGER	N	N	N	
	customerStatus	Status of customer	Varchar(10)	Y	N	N	Hoax
WalkInOrder	TimeWalkedIn	The time that the customer entered the shop	Time	Y	N	N	
PhoneOrder	TimeAnsweredCall	The time the phone call was answered	Time	Y	N	N	
	TimeTerminatedCall	The time the call was terminated	Time	Y	N	N	
PickUpOrder	pickUpTime	The time that the order was picked up from the store	Time	Y	N	N	

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Delivery Order	deliveryTime	The time that the order was delivered	Time	Y	N	N	
	deliveryAddress	The delivery address	Char(30)	Y	N	N	
	DriverStaffInfo	The details of the driver staff member that made the delivery	Char(20)	Y	N	N	
MenuItems	ItemCode	Unique MenuItems Identifier	Var Char(20)	N	N	N	
	Name	The name of the menu items	Char(20)	Y	N	N	
	PizzaSize	The size of the pizza	Char(20)	Y	N	N	
	CurrentSelling Price	The current selling price of the menu items	Float(5)	Y	N	N	
	Description	The description of the menu item	Char(50)	Y	N	N	

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Ingredient	Code	Unique ingredient identifier	Var char(20)	N	N	N	
	Name	Name of ingredient	Char(20)	Y	N	N	
	IngredientType	Type of ingredient (veg, meat ..etc)	Char(10)	Y	N	N	
	Description	Description of the ingredient	Char(50)	Y	N	N	
	StockLevelAtCurrentPeriod	The stock level at the current period for the ingredient	INTEGER	Y	N	N	
	StockLevelAtLastStockTake	The stock level at the last stock take of the ingredient	INTEGER	Y	N	N	
	SuggestedStockLevel	The suggested stock level	INTEGER	Y	N	N	
	ReorderLevel	The level at which more of the ingredient should be ordered	INTEGER	Y	N	N	
	DateLastStocktakeTaken	The last date the stocktake was taken	Date	Y	N	N	
	shelfLife	The shelf life of the ingredient	INTEGER	Y	N	N	

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Supplier	SupplierNo	Unique Supplier Identifier	Varchar (10)	N	N	N	
	Name	Name of supplier	Char (20)	Y	N	N	
	Address	Addresses of the supplier	Char (30)	Y	Y	N	
	Phone	The phone number of the contact person	Integer	Y	N	N	
	Email	Email address of the contact person	Char (50)	Y	N	N	
	ContactPerson	The contact person from the supplier	Char (20)	Y	N	N	
Ingredient Order	OrderNo	Unique Ingredient Order Identifier	Varchar (10)	N	N	N	
	DateOfOrder	Date of the order	Date	Y	N	N	
	DateRecieved Order	Date the order was received	Date	Y	N	N	
	TotalAmount	Total amount for the order	Float(5)	Y	N	N	
	OrderStatus	The order status	Varchar (10)	Y	N	N	
	Description	Description of the order	Varchar (50)	Y	N	N	

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Staff	EmployeeNumber	Unique Employee Identifier	VarChar (10)	N	N	N	
	FirstName	First name of the staff member	Char (10)	Y	N	N	
	LastName	First name of the staff member	Char (10)	Y	N	N	
	PostalAddress	Postal address	Char (50)	Y	N	N	
	ContactAddress	Contact address	Char (50)	Y	N	N	
	TaxFileNumber	Tax File number of the staff	Integer	Y	N	N	
	BankDetails	Bank details of staff	Varchar (50)	Y	Y	N	
	bankCode	Bank code of the staff member	Integer	Y	N	N	
	bankName	Bank name in which the staff member account exists	Char (30)	Y	N	N	
	bankAccountNumber	The account number of the staff members	Integer	Y	N	N	
	PaymentRate	The payment rate of the staff member	Float (10)	Y	N	N	
	workStatus	The current status of the staff (part time or full time)	Char (20)	Y	N	N	
	Description	A description of the staff	Char (50)	Y	N	N	

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
ShopStaff	shopRole	The role of a staff member in the shop (cashier, pizza maker etc.)	Char (50)	Y	N	N	
DriverStaff	LicenseNumber	The licence number of the delivery staff member	Integer	Y	N	N	
Shift	ShiftId	The shift identification number	Integer	N	N	N	
	StartDate	The start date of the shift	Date	Y	N	N	
	StartTime	The start time of the shift	Time	Y	N	N	
	EndDate	The end date of the shift	Date	Y	N	N	
	EndTime	The end time of the shift	Time	Y	N	N	
ShopStaffShift	BreakTime	The total time in which a shop staff member takes a break during a shift	Char(20)	Y	N	N	
DeliveryStaffShift	NumOfDeliveries	The number of deliveries that a delivery staff member completes in a shift	Integer	Y	N	N	1

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
Payment-Record	PaymentRecordID	Unique Payment Record Identifier	VarChar (10)	N	N	N	
	GrossPayment	Amount of money paid to employee	Float	Y	N	Y	
	TaxWithheld	The amount of money held back from the gross payment for tax	Float	Y	N	Y	
	TotalAmountPaid	Total amount paid to the employee	Float	Y	N	Y	
	PaymentDate	The day that the payment was made	Date	Y	N	N	
	PaymentPeriodStartDate	Start date of the payment period	Date	Y	N	N	
	PaymentPeriodEndDate	End date of the Payment Period	Date	Y	N	N	
	PaymentRate	The payment rate of the employee at the time of payment for the shift	Float	Y	N	N	

Entity	Attributes	Description	Data Type & Length	Nulls	Multi-valued	Derived	Default
DriverStaff Payment	TotalNumberOfDeliveries	The total number of deliveries made by a driver staff member during employment history	Integer	Y	N	N	
StoreStaff Payment	TotalNumberOfHoursWorked	The total number of hours worked by a shop staff member during employment history	Integer	Y	N	N	

Reflection on assignment 2 submission

Assignment 2 helped me understand the mapping of the EER model to the relational model in DBDL and normalizing the relational schema to Boyce-Codd Normal Form. Consulting with the tutor, certain changes to database design were discussed to further develop the design of the database. One change that was discussed was the addition of a 'PaymentRate' in the Payment Record object which would store the payment rate of an employee at the time of payment and employment.

Without the addition of the payment rate in payment record, if an employee received a payment rate rise in the future then it would result in the employee being supposedly underpaid on previous shifts. Other changes include changing the names of certain attribute names and datatype in some of the objects which could cause problems in the physical database design.

Relation Model from EER

To make marking this part easier, I've put each entity in its own box.

IngredientOrder (OrderNo, DateOfOrder, DateRecievedOrder, TotalAmount, OrderStatus, Description, SupplierNo)

Primary Key OrderNo

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

// the sum of the all the prices of each ingredient being ordered

---

Supplier (SupplierNo, Name, Address, Phone, Email, ContactPerson)

Primary Key SupplierNo

Ingredient (Code, Name, IngredientType, Description, StockLevelAtCurrentPeriod, StockLevelAtLastStocktake, SuggestedStockLevel, ReorderLevel, DateLastStocktakeTaken, shelfLife)

Primary Key Code

IsAPartOf (OrderNo, Code, quantity, price)

Primary Key OrderNo, Code

Foreign Key OrderNo references IngredientOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

Supplies (SupplierNo, Code, PriceOfIngredients, QuantityOfIngredients)

Primary Key SupplierNo, Code

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

MenuItems (ItemCode, Name, PizzaSize, CurrentSellingPrice, Description)

Primary Key ItemCode

Includes (Code, ItemCode, Quantity)

Primary Key Code, ItemCode

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ItemCode references MenuItems (ItemCode)

ON UPDATE CASCADE **ON DELETE** CASCADE

Has (OrderNo, ItemCode, Quantity, Price)

Primary Key OrderNo, ItemCode

Foreign Key OrderNo references WalkInOrder(OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key OrderNo references PickupOrder(OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key OrderNo references DeliveryOrder(OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ItemCode references MenuItems (ItemCode)

ON UPDATE CASCADE **ON DELETE** CASCADE

Customer (CusId, fName, lName, placeOfResident, Phone, Status)

Primary Key CusId

WalkinOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeWalkedIn)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

PickUpOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeAnsweredCall, TimeTerminatedCall, pickUpTime)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

DeliveryOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeAnsweredCall, TimeTerminatedCall, deliveryTime, deliveryAddress, DriverStaffInfo, ShiftId)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ShiftId references DeliveryStaffShift (ShiftId)

ON UPDATE CASCADE **ON DELETE** CASCADE

ShopStaff (EmployeeNumber, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankName, bankAccountNumber, PaymentRate, workStatus, Description, shopRole)

Primary Key EmployeeNumber

DriverStaff (EmployeeNumber, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankName, bankAccountNumber, PaymentRate, workStatus, Description, LicenseNumber)

Primary Key EmployeeNumber

ShopStaffShift (ShiftId, StartDate, StartTime, EndDate, EndTime, EmployeeNumber, BreakTime, ShopStaffEmployeeNumber, PaymentRecordId)

Primary Key ShiftId

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ShopStaffEmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key PaymentRecordId references StoreStaffPayment (PaymentRecordId)

ON UPDATE CASCADE **ON DELETE** CASCADE

DeliveryStaffShift (ShiftId, StartDate, StartTime, EndDate, EndTime, EmployeeNumber, NumOfDeliveries, DriveStaffEmployeeNumber, PaymentRecordId)

Primary Key ShiftId

Foreign Key EmployeeNumber references DriverStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key DriveStaffEmployeeNumber references DriverStaff(EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key PaymentRecordId references DriverStaffPayment(PaymentRecordId)

ON UPDATE CASCADE **ON DELETE** CASCADE

DriverStaffPayment (PaymentRecordId, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate, PaymentRate, EmployeeNumber, TotalNumberOfDeliveries)

Primary Key PaymentRecordId

Foreign Key EmployeeNumber references DriverStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived GrossPayment (totalnumberOfDelivers * PaymentRate)

Derived TaxWithheld (GrossPayment * 0.15)

Derived TotalAmountPaid (GrossPayment – TaxWithheld)

// assuming that tax is 15% of the gross payment

StoreStaffPayment (PaymentRecordId, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate, PaymentRate, EmployeeNumber, TotalNumberOfHoursWorked)

Primary Key PaymentRecordId

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived GrossPayment (hoursWorked * PaymentRate)

Derived TaxWithheld (GrossPayment * 0.15)

Derived TotalAmountPaid (GrossPayment – TaxWithheld)

// assuming that tax is 15% of the gross payment

IngredientOrder (OrderNo, DateOfOrder, DateRecievedOrder, TotalAmount, OrderStatus, Description, SupplierNo)

Primary Key OrderNo

Foreign Key SupplierNo references Supplier (SupplierNo)

// the sum of the all the prices of each ingredient being ordered

1NF

The entity is in first normal form as all attributes are in a single, atomic value from its domain. There is no attribute that is multiple or composite value.

2NF

The entity is in second normal form as it is already in 1st normal form and every non-candidate key attributes if fully functionally dependent on a candidate key.

3NF

The relation is in 3rd normal form as it is already in 2nd normal form and no non-candidate-key attribute is transitively dependent on a candidate key.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

IngredientOrder (OrderNo, DateOfOrder, DateRecievedOrder, TotalAmount, OrderStatus, Description, SupplierNo)

Primary Key OrderNo

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Supplier (<u>SupplierNo</u>, Name, Address, Phone, Email, ContactPerson)

Primary Key SupplierNo

1NF

The entity is not in 1st normal form as there is one attribute (address) that is multiple value. So we must transform the entity to first normal form.

<u>Functional Dependencies:</u>

FD1: SupplierNo → Name, Address, Phone, Email, ContactPerson

FD2: SupplierNo → address(address is a repeating group/not a single atomic value)

A new relation is created to store the addresses of the supplier:

Supplier (<u>SupplierNo</u>, Name, Phone, Email, ContactPerson)

Primary Key SupplierNo

Address (SupplierNo, Address)

Primary Key SupplierNo, Address

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

2NF

The 'supplier' entity is in 2nd normal form as the relation is in 1st normal form and every non-candidate key attribute is fully functionally dependent on a candidate key.

3NF

The relation is not in 3rd Normal Form as there is transitive dependencies.

Functional Dependencies:

<u>FD1:</u> SupplierNo → Name, Address, Phone, Email, ContactPerson

<u>FD2:</u> ContactPerson → Phone, Email

Phone and Email are redundant data as every time you have a contact person appearing, you will always have the same phone and email repeated. Hence they need to be separated into a different table. X = ContactPerson, Y = Phone, Email

<u>R – Y:</u>

Supplier (<u>SupplierNo</u>, Name, Address, ContactPerson)

XY:

ContactPerson (ContactPerson, Phone, Email)

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

Supplier (SupplierNo, Name, Address, Phone, Email, ContactPerson)

Primary Key SupplierNo

Foreign Key ContactPerson references ContactPerson (ContactPerson)

ON UPDATE CASCADE **ON DELETE** NO ACTION

ContactPerson (ContactPerson, Phone, Email)

Primary Key ContactPerson

Address (SupplierNo, Address)

Primary Key SupplierNo, Address

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Ingredient (Code, Name, IngredientType, Description, StockLevelAtCurrentPeriod, StockLevelAtLastStocktake, SuggestedStockLevel, ReorderLevel, DateLastStocktakeTaken, shelfLife)

Primary Key Code

1NF

The entity is in 1st normal form as all attributes are in a single, atomic value from its domain. There is no attribute value that is multiple or composite value.

2NF

The entity is in 2nd normal form as the relation is in 1st normal form and every non-candidate key attribute is fully functionally dependent on a candidate key.

3NF

The relation is not in 3rd Normal Form so we must transform it from 2nd Normal Form to 3rd Normal Form.

Functional Dependicies:

FD1: Code → Name, IngredientType, Description, StockLevelAtCurrentPeriod, StockLevelAtLastStocktake, SuggestedStockLevel, ReorderLevel, DateLastStocktakeTaken, shelfLife

FD2: IngredientType, → shelfLife

X → Y, where X = IngredientType,, Y = shelfLife

R – Y:

Ingredient (Code, Name, Description, StockLevelAtCurrentPeriod, StockLevelAtLastStocktake, SuggestedStockLevel, ReorderLevel, DateLastStocktakeTaken, IngredientType,)

XY:

Storage (IngredientType, shelfLife)

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result

Ingredient (Code, Name, Description, StockLevelAtCurrentPeriod, StockLevelAtLastStocktake, SuggestedStockLevel, ReorderLevel, DateLastStocktakeTaken, IngredientType)

Primary Key Code

Foreign Key IngredientType, references Storage (Type)

ON UPDATE CASCADE **ON DELETE** NO ACTION

Storage (IngredientType, shelfLife)

Primary Key IngredientType

IsAPartOf (OrderNo, Code, quantity, price)

Primary Key OrderNo, Code

Foreign Key OrderNo references IngredientOrder (OrderNo)

Foreign Key Code references Ingredient (Code)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

IsAPartOf (<u>OrderNo</u>, <u>Code</u>, quantity, price)

Primary Key OrderNo, Code

Foreign Key OrderNo references IngredientOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

Supplies (<u>SupplierNo</u>, <u>Code</u>, PriceOfIngredients, QuantityOfIngredients)

Primary Key SupplierNo, Code

Foreign Key SupplierNo references Supplier (SupplierNo)

Foreign Key Code references Ingredient (Code)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

Supplies (SupplierNo, Code, PriceOfIngredients, QuantityOfIngredients)

Primary Key SupplierNo, Code

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

MenuItems (ItemCode, Name, PizzaSize, CurrentSellingPrice, Description)

Primary Key ItemCode

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result

MenuItems (<u>ItemCode</u>, Name, PizzaSize, CurrentSellingPrice, Description)

Primary Key ItemCode

Includes (Code, ItemCode, Quantity)

Primary Key Code, ItemCode

Foreign Key Code references Ingredient (Code)

Foreign Key ItemCode references MenuItems (ItemCode)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

<u>Final Result:</u>

Includes (<u>Code</u>, <u>ItemCode</u>, Quantity)

Primary Key Code, ItemCode

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ItemCode references MenuItems (ItemCode)

ON UPDATE CASCADE **ON DELETE** CASCADE

Has (OrderNo, ItemCode, Quantity, Price)

Primary Key OrderNo, ItemCode

Foreign Key OrderNo references WalkInOrder (OrderNo)

Foreign Key OrderNo references PickupOrder (OrderNo)

Foreign Key OrderNo references DeliveryOrder (OrderNo)

Foreign Key ItemCode references MenuItems (ItemCode)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

<u>Final Result:</u>

Has (<u>OrderNo</u>, <u>ItemCode</u>, Quantity, Price)

Primary Key OrderNo, ItemCode

Foreign Key OrderNo references WalkInOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key OrderNo references PickupOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key OrderNo references DeliveryOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ItemCode references MenuItems (ItemCode)

ON UPDATE CASCADE **ON DELETE** CASCADE

Customer (<u>CusId</u>, fName, lName, placeOfResident, Phone, Status)

Primary Key CusId

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X $\rightarrow$ Y in R, X contains a candidate key

Final Result:

Customer (CusId, fName, lName, placeOfResident, Phone, Status)

Primary Key CusId

WalkinOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeWalkedIn)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

WalkinOrder (<u>OrderNo</u>, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeWalkedIn)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

PickUpOrder (<u>OrderNo</u>, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeAnsweredCall, TimeTerminatedCall, pickUpTime)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

PickUpOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeAnsweredCall, TimeTerminatedCall, pickUpTime)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

DeliveryOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeID, TimeAnsweredCall, TimeTerminatedCall, deliveryTime, deliveryAddress, DriverStaffInfo, ShiftId)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

Foreign Key ShiftId references DeliveryStaffShift (ShiftId)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

DeliveryOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeID, TimeAnsweredCall, TimeTerminatedCall, deliveryTime, deliveryAddress, DriverStaffInfo, ShiftId)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references DriverStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ShiftId references DeliveryStaffShift (ShiftId)

ON UPDATE CASCADE **ON DELETE** CASCADE

ShopStaff (EmployeeNumber, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankName, bankAccountNumber, PaymentRate, workStatus, Description, shopRole)

Primary Key EmployeeNumber

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is not is 3rd Normal Form. So we must transform it into 3rd normal form.

Functional Dependencies:

FD1:

EmployeeNumber → FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankAccountNumber, PaymentRate, workStatus, Description, shopRole, bankCode (Primary Key)

FD2:

bankCode → bankName (Transitive dependency)

Decomposing ShopStaff relation based on FD2:

X → Y, where X = bankCode, Y = bankName

R – Y:

ShopStaff (EmployeeNumber, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankAccountNumber, PaymentRate, workStatus, Description, shopRole)

XY:

Bank (bankCode, bankName)

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result

ShopStaff (EmployeeNumber, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankAccountNumber, PaymentRate, workStatus, Description, shopRole)

Primary Key EmployeeNumber

Foreign Key bankCode references Bank(bankCode)

ON UPDATE CASCADE **ON DELETE** NO ACTION

Bank (bankCode, bankName)

Primary Key bankCode

DriverStaff (EmployeeNumber, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankName, bankAccountNumber, PaymentRate, workStatus, Description, LicenseNumber)

Primary Key EmployeeNumber

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and every non-candidate key attribute is full functionally dependent on any candidate key.

3NF

The relation is not is 3rd Normal Form. So we must transform it into 3rd normal form.

<u>Functional Dependencies:</u>

<u>FD1:</u>

EmployeeNumber → FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankName, bankAccountNumber, PaymentRate, workStatus, Description, LicenseNumber, bankCode (Primary Key)

<u>FD2:</u>

bankCode → bankName (Transitive dependency)

Decomposing ShopStaff relation based on FD2:

X → Y, where X = bankCode, Y = bankName

<u>R – Y:</u>

DriverStaff (EmployeeNumber, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankAccountNumber, PaymentRate, workStatus, Description, LicenseNumber)

<u>XY:</u>

Bank (<u>bankCode</u>, bankName)

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

<u>Final Result:</u>

DriverStaff (<u>EmployeeNumber</u>, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankAccountNumber, PaymentRate, workStatus, Description, LicenseNumber)

Primary Key EmployeeNumber

Foreign Key bankCode references Bank(bankCode)

ON UPDATE CASCADE **ON DELETE** NO ACTION

Bank (<u>bankCode</u>, bankName)

Primary Key bankCode

ShopStaffShift (ShiftId, StartDate, StartTime, EndDate, EndTime, EmployeeNumber, BreakTime, ShopStaffEmployeeNumber, PaymentRecordId)

Primary Key ShiftId

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

Foreign Key ShopStaffEmployeeNumber references ShopStaff (EmployeeNumber)

Foreign Key PaymentRecordId references StoreStaffPayment (PaymentRecordId)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

ShopStaffShift (ShiftId, StartDate, StartTime, EndDate, EndTime, EmployeeNumber, BreakTime, ShopStaffEmployeeNumber, PaymentRecordId)

Primary Key ShiftId

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ShopStaffEmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key PaymentRecordId references StoreStaffPayment (PaymentRecordId)

ON UPDATE CASCADE **ON DELETE** CASCADE

DeliveryStaffShift (ShiftId, StartDate, StartTime, EndDate, EndTime, EmployeeNumber, NumOfDeliveries, DriveStaffEmployeeNumber, PaymentRecordId)

Primary Key ShiftId

Foreign Key EmployeeNumber references DriverStaff (EmployeeNumber)

Foreign Key DriveStaffEmployeeNumber references DriverStaff (EmployeeNumber)

Foreign Key PaymentRecordId references DriverStaffPayment (PaymentRecordId)

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X → Y in R, X contains a candidate key

Final Result:

DeliveryStaffShift (ShiftId, StartDate, StartTime, EndDate, EndTime, EmployeeNumber, NumOfDeliveries, DriveStaffEmployeeNumber, PaymentRecordId)

Primary Key ShiftId

Foreign Key EmployeeNumber references DriverStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key DriveStaffEmployeeNumber references DriverStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key PaymentRecordId references DriverStaffPayment (PaymentRecordId)

ON UPDATE CASCADE **ON DELETE** CASCADE

DriverStaffPayment (PaymentRecordId, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate, PaymentRate, EmployeeNumber, TotalNumberOfDeliveries)

Primary Key PaymentRecordId

Foreign Key EmployeeNumber references Shop Staff (EmployeeNumber)

Derived GrossPayment (totalnumberOfDelivers* PaymentRate)

Derived TaxWithheld (GrossPayment * 0.15)

Derived TotalAmountPaid (GrossPayment – TaxWithheld)

// assuming that tax is 15% of the gross payment

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, $X \rightarrow Y$ in R, X contains a candidate key

Final Result :

DriverStaffPayment (PaymentRecordId, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate, PaymentRate, EmployeeNumber, TotalNumberOfDeliveries)

Primary Key PaymentRecordId

Foreign Key EmployeeNumber references Shop Staff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived GrossPayment (totalnumberOfDelivers* PaymentRate)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived TaxWithheld (GrossPayment * 0.15)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived TotalAmountPaid (GrossPayment – TaxWithheld)

ON UPDATE CASCADE **ON DELETE** CASCADE

StoreStaffPayment (<u>PaymentRecordId</u>, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate, PaymentRate, EmployeeNumber, TotalNumberOfHoursWorked)

Primary Key PaymentRecordId

Foreign Key EmployeeNumber references DriverStaff (EmployeeNumber)

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

Derived GrossPayment (TotalNumberOfHoursWorked * PaymentRate)

Derived TaxWithheld (GrossPayment * 0.15)

Derived TotalAmountPaid (GrossPayment – TaxWithheld)

// assuming that tax is 15% of the gross payment

1NF

The relation is in 1st Normal Form as each attribute is a single, atomic value from its domain.

2NF

The relation is in 2nd Normal Form as it is already in 1st Normal Form and there is no partial dependencies in the relation table.

3NF

The relation is in 3rd Normal Form as it is already in 2nd Normal Form and there is no transitive dependencies in the relation table.

BCNF

The relation is in BCNF as it satisfies the condition:

Every functional dependency, X $\rightarrow$ Y in R, X contains a candidate key

<u>Final Result:</u>

StoreStaffPayment (<u>PaymentRecordId</u>, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate, PaymentRate, EmployeeNumber, TotalNumberOfHoursWorked)

Primary Key PaymentRecordId

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived GrossPayment (TotalNumberOfHoursWorked * PaymentRate)

Derived TaxWithheld (GrossPayment * 0.15)

Derived TotalAmountPaid (GrossPayment – TaxWithheld)

<u>Summarised BCNF entities:</u>

IngredientOrder (<u>OrderNo</u>, DateOfOrder, DateRecievedOrder, OrderStatus, Description, SupplierNo)

Primary Key OrderNo

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Supplier (<u>SupplierNo</u>, Name, Phone, Email, ContactPerson)

Primary Key SupplierNo

Foreign Key ContactPerson references ContactPerson (ContactPerson)

ON UPDATE CASCADE **ON DELETE** NO ACTION

ContactPerson (<u>ContactPerson</u>, Phone, Email)

Primary Key ContactPerson

Address (SupplierNo, Address)

Primary Key SupplierNo, Address

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Ingredient (Code, Name, Description, StockLevelAtCurrentPeriod, StockLevelAtLastStocktake, SuggestedStockLevel, ReorderLevel, DateLastStocktakeTaken, IngredientType)

Primary Key Code

Foreign Key IngredientType references Storage (IngredientType)

ON UPDATE CASCADE **ON DELETE** NO ACTION

Storage (IngredientType, shelfLife)

Primary Key IngredientType

IsAPartOf (OrderNo, Code, quantity, price)

Primary Key OrderNo, Code

Foreign Key OrderNo references IngredientOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

Supplies (SupplierNo, Code, PriceOfIngredients, QuantityOfIngredients)

Primary Key SupplierNo, Code

Foreign Key SupplierNo references Supplier (SupplierNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

MenuItems (ItemCode, Name, PizzaSize, CurrentSellingPrice, Description)

Primary Key ItemCode

Includes (Code, ItemCode, Quantity)

Primary Key Code, ItemCode

Foreign Key Code references Ingredient (Code)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ItemCode references MenuItems (ItemCode)

ON UPDATE CASCADE **ON DELETE** CASCADE

Has (<u>OrderNo</u>, <u>ItemCode</u>, Quantity, Price)

Primary Key OrderNo, ItemCode

Foreign Key OrderNo references WalkInOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key OrderNo references PickupOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key OrderNo references DeliveryOrder (OrderNo)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ItemCode references MenuItems (ItemCode)

ON UPDATE CASCADE **ON DELETE** CASCADE

Customer (<u>CusId</u>, fName, lName, placeOfResident, Phone, Status)

Primary Key CusId

WalkinOrder (<u>OrderNo</u>, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeWalkedIn)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

PickUpOrder (<u>OrderNo</u>, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeNumber, TimeAnsweredCall, TimeTerminatedCall, pickUpTime)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

DeliveryOrder (<u>OrderNo</u>, OrderDateTime, TotalAmountDue, PaymentMethod, paymentApprovalNo, OrderStatus, Description, CusId, EmployeeID, TimeAnsweredCall, TimeTerminatedCall, deliveryTime, deliveryAddress, DriverStaffInfo, ShiftId)

Primary Key OrderNo

Foreign Key CusID references Customer (CusID)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ShiftId references DeliveryStaffShift (ShiftId)

ON UPDATE CASCADE **ON DELETE** CASCADE

ShopStaff (<u>EmployeeNumber</u>, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankAccountNumber, PaymentRate, workStatus, Description, shopRole)

Primary Key EmployeeNumber

Foreign Key bankCode references Bank(bankCode)

ON UPDATE CASCADE **ON DELETE** NO ACTION

Bank (<u>bankCode</u>, bankName)

Primary Key bankCode

DriverStaff (<u>EmployeeNumber</u>, FirstName, LastName, PostalAddress, ContactAddress, TaxFileNumber, bankCode, bankAccountNumber, PaymentRate, workStatus, Description, LicenseNumber)

Primary Key EmployeeNumber

Foreign Key bankCode references Bank(bankCode)

ON UPDATE CASCADE **ON DELETE** NO ACTION

Bank (<u>bankCode</u>, bankName)

Primary Key bankCode

ShopStaffShift (ShiftId, StartDate, StartTime, EndDate, EndTime, EmployeeNumber, BreakTime, ShopStaffEmployeeNumber, PaymentRecordId)

Primary Key ShiftId

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key ShopStaffEmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key PaymentRecordId references StoreStaffPayment (PaymentRecordId)

ON UPDATE CASCADE **ON DELETE** CASCADE

DeliveryStaffShift (<u>ShiftId</u>, StartDate, StartTime, EndDate, EndTime, EmployeeNumber, NumOfDeliveries, DriveStaffEmployeeNumber, PaymentRecordId)

Primary Key ShiftId

Foreign Key EmployeeNumber references DriverStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key DriveStaffEmployeeNumber references DriverStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Foreign Key PaymentRecordId references DriverStaffPayment (PaymentRecordId)

ON UPDATE CASCADE **ON DELETE** CASCADE

DriverStaffPayment (<u>PaymentRecordId</u>, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate, PaymentRate, EmployeeNumber, TotalNumberOfDeliveries)

Primary Key PaymentRecordId

Foreign Key EmployeeNumber references DriverStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived GrossPayment (totalnumberOfDelivers* PaymentRate)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived TaxWithheld (GrossPayment * 0.15)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived TotalAmountPaid (GrossPayment – TaxWithheld)

ON UPDATE CASCADE **ON DELETE** CASCADE

StoreStaffPayment (<u>PaymentRecordId</u>, PaymentDate, PaymentPeriodStartDate, PaymentPeriodEndDate, PaymentRate, EmployeeNumber, TotalNumberOfHoursWorked)

Primary Key PaymentRecordId

Foreign Key EmployeeNumber references ShopStaff (EmployeeNumber)

ON UPDATE CASCADE **ON DELETE** CASCADE

Derived GrossPayment (TotalNumberOfHoursWorked * PaymentRate)

Derived TaxWithheld (GrossPayment * 0.15)

Derived TotalAmountPaid (GrossPayment – TaxWithheld)

Implementing the Physical Database Design

After a discussion with the tutor, the relationship between PaymentRecord to Staff and Staff to Shift was removed when implementing the normalized into T-SQL as it introduced redundant foreign keys. For example, DriverPaymentRecord has a foreign key to the DriverStaff entity through the relationship between PaymentRecord and Staff in the EER. As well as a foreign key that it receives through the relationship between itself and the DeliveryStaffShift entity.

So it was reasonable to remove the relationships as it would also avoid multiple cascading paths and redundant foreign keys. Certain attribute names needed to be changed when implementing the database design into SQL. I changed the attributes name through the EER and this document. Apologies if there some places where the attributes still have their old name.

```
DROP TABLE HasDelivery;

DROP TABLE HasPickUp;

DROP TABLE HasWalkIn;

DROP TABLE includes;

DROP TABLE Supplies;

DROP TABLE IsAPartOf;

DROP TABLE ShopStaffShift;

DROP TABLE StoreStaffPayment;

DROP TABLE DeliveryOrder;

DROP TABLE DeliveryStaffShift;

DROP TABLE DriverStaffPayment;

DROP TABLE PickUpOrder;

DROP TABLE WalkInOrder;

DROP TABLE DriverStaff;

DROP TABLE ShopStaff;

DROP TABLE Bank;

DROP TABLE Customer;

DROP TABLE IngredientOrder;

DROP TABLE Address;

DROP TABLE Supplier;

DROP TABLE ContactPerson;

DROP TABLE Ingredient;

DROP TABLE Storage;

DROP TABLE MenuItems;


CREATE TABLE MenuItems(

        ItemCode VARCHAR(20) NOT NULL PRIMARY KEY,

        Name CHAR(20) ,

        PizzaSize CHAR(20),

        CurrentSellingPrice FLOAT(5),

        Description CHAR(50),
        CHECK(PizzaSize IN ('Small', 'Medium', 'Large'))

);

INSERT INTO MenuItems VALUES (001, 'Spicy Trio' , 'Large', 12, 'Vegetarian Pizza');
INSERT INTO MenuItems VALUES (002, 'Pineapple Crusty' , 'Small', 16, 'Vegetarian Pizza
with Pineapple');
INSERT INTO MenuItems VALUES (003, 'Flaming Chillies' , 'Medium', 13, 'Cheese Pizza');
INSERT INTO MenuItems VALUES (004, 'New Yorker' , 'Medium', 13, 'Famous New Yorker
Pizza');

--------------------------------------------------------------------------------------------------------------

CREATE TABLE Storage(
        IngredientType CHAR(10) NOT NULL PRIMARY KEY,
        shelfLife INTEGER
);

INSERT INTO Storage VALUES ('Veg', 14);
INSERT INTO Storage VALUES ('Meat', 5);
INSERT INTO Storage VALUES ('Fruit', 7);

CREATE TABLE Ingredient(
        Code VARCHAR(20) NOT NULL PRIMARY KEY,
        Name CHAR(20),
        IngredientType CHAR(10),
        Description CHAR(50),
        StockLevelCurrentPeriod INTEGER,
        StockLevelLastStockTake INTEGER,
        SuggestedStockLevel INTEGER,
        ReorderLevel INTEGER,
        DateLastStocktakeTaken DATE,
	FOREIGN KEY (IngredientType) REFERENCES Storage(IngredientType) ON
UPDATE CASCADE ON DELETE CASCADE

);


INSERT INTO Ingredient VALUES (1, 'Pineapple', 'Fruit', 'Chopped Pineapple', 15, 17, 13,
10, CONVERT(date, 'Oct 12 2019'));

INSERT INTO Ingredient VALUES (2, 'Chicken', 'Meat', 'Fine Chopped Chicken Breast
Pieces', 13, 15, 10, 8, CONVERT(date, 'Oct 7 2019'));

INSERT INTO Ingredient VALUES (3, 'Mushrooms', 'Veg', 'Thin slices of Mushroom', 16, 18,
12, 9, CONVERT(date, 'Oct 21 2019'));

INSERT INTO Ingredient VALUES (4, 'Chilly Flakes', 'Veg', 'Finely Chopped Chilly', 16, 18,
12, 9, CONVERT(date, 'Oct 23 2019'));


----------------------------------------------------------------------------------------------------------------


CREATE TABLE ContactPerson (

	ContactPerson CHAR(20) NOT NULL PRIMARY KEY,

	Phone INTEGER,

	Email CHAR(50),

);


INSERT INTO ContactPerson VALUES ('James', 0491570156, 'James@gmail.com');

INSERT INTO ContactPerson VALUES ('Kevin', 0491570157, 'Kevin@gmail.com');

INSERT INTO ContactPerson VALUES ('Steve', 0491570158, 'Steve@yahoo.com');


CREATE TABLE Supplier(

	SupplierNo VARCHAR(10) NOT NULL PRIMARY KEY,

	Name CHAR(20),

	ContactPerson CHAR(20),

	FOREIGN KEY (ContactPerson) REFERENCES ContactPerson(ContactPerson) ON
UPDATE CASCADE ON DELETE CASCADE

);


INSERT INTO Supplier VALUES (007, 'James Producers', 'James');
INSERT INTO Supplier VALUES (70, 'Big Kevs Veggys', 'Kevin');
INSERT INTO Supplier VALUES (21, 'Steve Productions', 'Steve');

CREATE TABLE Address(
        SupplierNo VARCHAR(10) NOT NULL,
        Address CHAR(30) NOT NULL,
        PRIMARY KEY (SupplierNo, Address),
        FOREIGN KEY (SupplierNo) REFERENCES Supplier(SupplierNo)
);

INSERT INTO Address VALUES (007, '21 Jump Street');
INSERT INTO Address VALUES (007, '23 Jump Street');
INSERT INTO Address VALUES (70, '22 Jump Street');
INSERT INTO Address VALUES (21,'5th Fifth Avenue');

-----------------------------------------------------------------------------------------------------------------

CREATE TABLE IngredientOrder(
        OrderNo VARCHAR(10) NOT NULL PRIMARY KEY,
        DateOfOrder DATE,
        DateReviecedOrder DATE,
        TotalAmount FLOAT(5),
        OrderStatus VARCHAR(10),
        Description VARCHAR(50),
        SupplierNo VARCHAR(10),
        FOREIGN KEY (SupplierNo) REFERENCES Supplier(SupplierNo) ON UPDATE
CASCADE ON DELETE CASCADE
);


INSERT INTO IngredientOrder VALUES('200',CONVERT(date, 'Oct 8 2019'),
CONVERT(date, 'Oct 13 2019'), 300,'Completed', 'ordering chicken', 007 );
```

```
INSERT INTO IngredientOrder VALUES('201',CONVERT(date, 'Oct 21 2019'),
CONVERT(date, 'Oct 26 2019'), 400, 'Recieved', 'ordering veggies' ,70);

INSERT INTO IngredientOrder VALUES('202',CONVERT(date, 'Oct 25 2019'),
CONVERT(date, 'Oct 30 2019'), 600, 'Recieved', 'ordering Mushroom' ,21);

INSERT INTO IngredientOrder VALUES('203',CONVERT(date, 'Oct 27 2019'),
CONVERT(date, 'Oct 30 2019'), 500, 'Recieved', 'ordering Chilly Flakes' ,21);


-------------------------------------------------------------------------------------------------------------


CREATE TABLE Customer (

        CusId CHAR(10) NOT NULL PRIMARY KEY,

        fName CHAR(15),

        lName CHAR(15),

        placeOfResident CHAR(50),

        Phone INTEGER,

        customerStatus VARCHAR(10) DEFAULT 'Hoax',

        CHECK(customerStatus IN ('Hoax', 'Verified'))

);


INSERT INTO Customer (CusId, fName,lName,placeOfResident, Phone) VALUES ('901',
'Anastasia', 'Page', '1710 Kelly Drive', 041234567);

INSERT INTO Customer VALUES ('902', 'Ryan', 'Goff', '4762 Coal Road', 047654321,
'Verified');

INSERT INTO Customer VALUES ('903', 'Harrison', 'Hope', '3940 Ashmor Drive',
047654123, 'Verified');


-------------------------------------------------------------------------------------------------------------


CREATE TABLE Bank(

        bankCode INTEGER NOT NULL PRIMARY KEY,

        bankName CHAR(30)

);


INSERT INTO Bank VALUES (123, 'ANZ');
INSERT INTO Bank VALUES (132, 'Commonwealth');

INSERT INTO Bank VALUES (321, 'NAB');


CREATE TABLE ShopStaff (

        EmployeeNumber VARCHAR(10) NOT NULL PRIMARY KEY,

        FirstName CHAR(10),

        LastName CHAR(10),

        PostalAddress CHAR(50),

        ContactAddress CHAR(50),

        TaxFileNumber INTEGER,

        bankCode INTEGER,

        bankAccountNumber INTEGER,

        PaymentRate FLOAT(10),

        workStatus CHAR(20),

        CHECK(workStatus IN ('Full Time', 'Part Time')),

        Description CHAR(50),

        shopRole CHAR(50),

        FOREIGN KEY (bankCode) REFERENCES Bank(bankCode) ON UPDATE
CASCADE ON DELETE CASCADE

);


INSERT INTO ShopStaff VALUES (101, 'Meera', 'Lister', '4421 Pine Tree Lane', '4421 Pine
Tree Lane', 123456789, 123, 70, 25.6, 'Full Time', 'Employee for 10 years', 'Cashier');

INSERT INTO ShopStaff VALUES (102, 'Vlad', 'Berry', '657 Half AND Half Drive', '2503
Gerald L. Bates Drive', 123456788, 132, 71, 30, 'Full Time', 'Employee for 4 years',
'Accountant');

INSERT INTO ShopStaff VALUES (103, 'Sadia', 'Deacon', '4572 Bassell Avenue', '4572
Bassell Avenue', 123456787, 321, 50, 35, 'Part Time', 'Employee for 7 years', 'Manager');


CREATE TABLE DriverStaff (

        EmployeeNumber VARCHAR(10) NOT NULL PRIMARY KEY,

        FirstName CHAR(10),

        LastName CHAR(10),

        PostalAddress CHAR(50),
        ContactAddress CHAR(50),

        TaxFileNumber INTEGER,

        bankCode INTEGER,

        bankAccountNumber INTEGER,

        PaymentRate FLOAT(10),

        workStatus CHAR(20),

        CHECK(workStatus IN ('Full Time', 'Part Time')),

        Description CHAR(50),

        LicenseNumber INTEGER,

        FOREIGN KEY (bankCode) REFERENCES Bank(bankCode)

);


INSERT INTO DriverStaff VALUES (123, 'Codey', 'Price', '2769 Saint Marys Avenue', '2769
Saint Marys Avenue', 123456778, 123, 10, 23, 'Part Time', 'Drives subaru', 789);

INSERT INTO DriverStaff VALUES (124, 'Vlad', 'Berry', '2589 Williams Lane', '4903 Jerry
Toth Drive', 123456775, 123, 11, 24, 'Full Time', 'Drives toyota', 987);

INSERT INTO DriverStaff VALUES (125, 'Kevin', 'Deacon', '2832 Murphy Court', '2832
Murphy Court', 123456772, 321, 12, 22, 'Part Time', 'Drives mazda', 897);


----------------------------------------------------------------------------------------------------------------


CREATE TABLE WalkInOrder(

        OrderNo CHAR(10) NOT NULL PRIMARY KEY,

        OrderDateTime DATETIME,

        TotalAmountDue FLOAT,

        PaymentMethod VARCHAR(20),

        CHECK(PaymentMethod IN ('Credit Card', 'Debit Card', 'Cash')),

        paymentApprovalNo CHAR(10) DEFAULT NULL,

        OrderStatus VARCHAR(10),

        Description CHAR(30),

        CusId CHAR(10),

        EmployeeNumber VARCHAR(10),

        TimeWalkedIn TIME,
        FOREIGN KEY (CusID) REFERENCES Customer(CusID),
        FOREIGN Key (EmployeeNumber) references ShopStaff (EmployeeNumber) ON
UPDATE CASCADE ON DELETE CASCADE
);


INSERT INTO WalkInOrder VALUES (60, CONVERT(datetime, 'Oct 13 2019 10:40:45 AM'),
200, 'Credit Card', '123456789', 'Completed', 'ordered 12 pizzas', '901', 101,
CONVERT(time, '10:35:00 AM') );
INSERT INTO WalkInOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod,
OrderStatus, Description, CusId, EmployeeNumber, TimeWalkedIn) VALUES (61,
CONVERT(datetime, 'Oct 17 2019 11:25:21 AM'), 300,  'Cash', 'Completed', 'ordered 10
pizzas', '902', 102, CONVERT(time, '11:35:00 AM') );
INSERT INTO WalkInOrder VALUES (62, CONVERT(datetime, 'Oct 20 2019 8:30:15 PM') ,
400, 'Credit Card', '123456666', 'Completed', 'ordered 7 pizzas', '903', 103, CONVERT(time,
'8:40:32 PM') );


-------------------------------------------------------------------------------------------------------------------


CREATE TABLE PickUpOrder (
        OrderNo CHAR(10) NOT NULL PRIMARY KEY,
        OrderDateTime DATETIME,
        TotalAmountDue FLOAT,
        PaymentMethod VARCHAR(20),
        CHECK(PaymentMethod IN ('Credit Card', 'Debit Card', 'Cash')),
        paymentApprovalNo CHAR(10) DEFAULT NULL,
        OrderStatus VARCHAR(10),
        Description CHAR(30),
        CusId CHAR(10),
        EmployeeNumber VARCHAR(10),
        TimeAnsweredCall TIME,
        TimeTerminatedCall TIME,
        pickUpTime TIME,
        FOREIGN KEY (CusID) REFERENCES Customer(CusID),
        FOREIGN Key (EmployeeNumber) references ShopStaff (EmployeeNumber) ON
UPDATE CASCADE ON DELETE CASCADE
```

```
);


INSERT INTO PickUpOrder VALUES (170, CONVERT(datetime, 'Oct 26 2019 4:00:00 PM'), 300, 'Credit Card', '500', 'Completed', 'order 3 pizzas', '901', 101, CONVERT(time, '3:59:00 PM'), CONVERT(time, '4:02:00 PM'), CONVERT(time, '4:30:00 PM'));

INSERT INTO PickUpOrder VALUES (171, CONVERT(datetime, 'Oct 26 2019 6:00:00 PM'), 400, 'Credit Card', '501', 'Completed', 'order 4 pizzas', '903', 101, CONVERT(time, '5:59:00 PM'), CONVERT(time, '6:02:00 PM'), CONVERT(time, '6:30:00 PM'));

INSERT INTO PickUpOrder (OrderNo, OrderDateTime, TotalAmountDue, PaymentMethod, OrderStatus, Description, CusId, EmployeeNumber, TimeAnsweredCall, TimeTerminatedCall, pickUpTime) VALUES (172, CONVERT(datetime, 'Oct 26 2019 8:00:00 PM'), 450, 'Cash', 'Completed', 'order 6 pizzas', '902', 103, CONVERT(time, '7:59:00 PM'), CONVERT(time, '8:02:00 PM'), CONVERT(time, '8:30:00 PM'))


-----------------------------------------------------------------------------------------------------------------


CREATE TABLE DriverStaffPayment(

        PaymentRecordId VARCHAR(10) NOT NULL PRIMARY KEY,

        PaymentDate DATE,

        PaymentPeriodStartDate DATE,

        PaymentPeriodEndDate DATE,

        PaymentRate FLOAT,

        TotalNumberOfDeliveries INTEGER

);


ALTER TABLE DriverStaffPayment ADD GrossPayment AS (TotalNumberOfDeliveries * PaymentRate);

ALTER TABLE DriverStaffPayment ADD TaxWitheld AS ((TotalNumberOfDeliveries * PaymentRate) * 0.15);

ALTER TABLE DriverStaffPayment ADD TotalAmountPaid AS ((TotalNumberOfDeliveries * PaymentRate) - ((TotalNumberOfDeliveries * PaymentRate) * 0.15));

GO


INSERT INTO DriverStaffPayment VALUES (1, CONVERT(date, 'Oct 31 2019'), CONVERT(date, 'Oct 17 2019'), CONVERT(date, 'Oct 31 2019'), 30, 10);

INSERT INTO DriverStaffPayment VALUES (2, CONVERT(date, 'Oct 31 2019'), CONVERT(date, 'Oct 17 2019'), CONVERT(date, 'Oct 31 2019'), 40, 5);
INSERT INTO DriverStaffPayment VALUES (3, CONVERT(date, 'Oct 31 2019'),
CONVERT(date, 'Oct 17 2019'), CONVERT(date, 'Oct 31 2019'), 15, 24);

INSERT INTO DriverStaffPayment VALUES (4, CONVERT(date, 'Oct 31 2019'),
CONVERT(date, 'Oct 17 2019'), CONVERT(date, 'Oct 31 2019'), 18, 10);
```

```
CREATE TABLE DeliveryStaffShift(

        ShiftId INTEGER NOT NULL PRIMARY KEY,

        StartDate DATE,

        StartTime TIME,

        EndDate DATE,

        EndTime TIME,

        EmployeeNumber VARCHAR(10),

        NumOfDeliveries INTEGER DEFAULT 1,

        PaymentRecordId VARCHAR(10),

   FOREIGN Key (EmployeeNumber) references DriverStaff(EmployeeNumber) ON
UPDATE CASCADE ON DELETE CASCADE,

        FOREIGN KEY (PaymentRecordId) REFERENCES
DriverStaffPayment(PaymentRecordId) ON UPDATE CASCADE ON DELETE CASCADE

        );


INSERT INTO DeliveryStaffShift VALUES (1, CONVERT(date, 'Oct 27 2019'),
CONVERT(time, '9:00:00 AM'), CONVERT(date, 'Oct 27 2019'), CONVERT(time, '3:00:00
PM'), 123, 12, 1);

INSERT INTO DeliveryStaffShift VALUES (2, CONVERT(date, 'Oct 27 2019'),
CONVERT(time, '10:00:00 AM'), CONVERT(date, 'Oct 27 2019'), CONVERT(time, '5:00:00
PM'), 124, 5, 2);

INSERT INTO DeliveryStaffShift VALUES (3, CONVERT(date, 'Oct 27 2019'),
CONVERT(time, '12:00:00 AM'), CONVERT(date, 'Oct 27 2019'), CONVERT(time, '8:00:00
PM'), 125, 24, 3);

INSERT INTO DeliveryStaffShift VALUES (4, CONVERT(date, 'Oct 18 2019'),
CONVERT(time, '10:00:00 AM'), CONVERT(date, 'Oct 18 2019'), CONVERT(time, '4:00:00
PM'), 123, 10, 4);
```

```
CREATE TABLE DeliveryOrder (
        OrderNo CHAR(10) NOT NULL PRIMARY KEY,
        OrderDateTime DATETIME,
        TotalAmountDue FLOAT,
        PaymentMethod VARCHAR(20),
        CHECK(PaymentMethod IN ('Credit Card', 'Debit Card', 'Cash')),
        paymentApprovalNo CHAR(10) DEFAULT NULL,
        OrderStatus VARCHAR(10),
        Description CHAR(30),
        CusId CHAR(10),
        EmployeeNumber VARCHAR(10),
        TimeAnsweredCall TIME,
        TimeTerminatedCall TIME,
        deliveryTime TIME,
        deliveryAddress CHAR(30),
        DriverStaffInfo CHAR(20),
        ShiftId INTEGER,
        FOREIGN KEY (CusID) REFERENCES Customer(CusID)  ON UPDATE CASCADE
ON DELETE CASCADE,
        FOREIGN Key (EmployeeNumber) references ShopStaff (EmployeeNumber) ON
UPDATE CASCADE ON DELETE CASCADE,
        FOREIGN Key (ShiftId) references DeliveryStaffShift(ShiftId) ON UPDATE
CASCADE ON DELETE CASCADE
);


INSERT INTO DeliveryOrder VALUES (1,
CONVERT(DATETIME, 'Oct 27 2019 11:50:00 AM'), 400, 'Credit Card', '1234567',
'delivered', '12 pizzas', '901', 101,
CONVERT(TIME, '11:30:00 AM'), CONVERT(TIME, '11:35:00 AM'), CONVERT(TIME,
'11:59:00 AM'), '21 Alton Rd', 'Driver 123', 1);


INSERT INTO DeliveryOrder VALUES (2,
CONVERT(DATETIME, 'Oct 27 2019 11:50:00 AM'), 450, 'Debit Card', '1234567', 'delivered',
'12 pizzas', '902', 102,
CONVERT(TIME, '11:30:00 AM'), CONVERT(TIME, '11:35:00 AM'), CONVERT(TIME,
'11:59:00 AM'), '9 Eleven Rd', 'Driver 124', 2);


INSERT INTO DeliveryOrder VALUES (3,

CONVERT(DATETIME, 'Oct 27 2019 11:50:00 AM'), 525, 'Debit Card', '1234567', 'delivered',
'12 pizzas', '903', 103,

CONVERT(TIME, '11:30:00 AM'), CONVERT(TIME, '11:35:00 AM'), CONVERT(TIME,
'11:59:00 AM'), '7 Ring Rd', 'Driver 124', 3);


-----------------------------------------------------------------------------------------------------------------


CREATE TABLE StoreStaffPayment(

        PaymentRecordId VARCHAR(10) NOT NULL PRIMARY KEY,

        PaymentDate DATE,

        PaymentPeriodStartDate DATE,

        PaymentPeriodEndDate DATE,

        PaymentRate FLOAT,

        TotalNumberOfHoursWorked INTEGER,

);


ALTER TABLE StoreStaffPayment ADD GrossPayment AS (TotalNumberOfHoursWorked *
PaymentRate);

ALTER TABLE StoreStaffPayment ADD TaxWitheld AS ((TotalNumberOfHoursWorked *
PaymentRate) * 0.15);

ALTER TABLE StoreStaffPayment ADD TotalAmountPaid AS
((TotalNumberOfHoursWorked * PaymentRate) - ((TotalNumberOfHoursWorked *
PaymentRate) * 0.15));

GO


INSERT INTO StoreStaffPayment VALUES (1, CONVERT(date, 'Oct 31 2019'),
CONVERT(date, 'Oct 17 2019'), CONVERT(date, 'Oct 31 2019'), 26.25, 12);

INSERT INTO StoreStaffPayment VALUES (2, CONVERT(date, 'Oct 31 2019'),
CONVERT(date, 'Oct 17 2019'), CONVERT(date, 'Oct 31 2019'), 25, 9);

INSERT INTO StoreStaffPayment VALUES (3, CONVERT(date, 'Oct 31 2019'),
CONVERT(date, 'Oct 17 2019'), CONVERT(date, 'Oct 31 2019'), 45, 10);
```

```
CREATE TABLE ShopStaffShift(
	ShiftId INTEGER NOT NULL PRIMARY KEY,
	StartDate DATE,
	StartTime TIME,
	EndDate DATE,
	EndTime TIME,
	EmployeeNumber VARCHAR(10),
	BreakTime CHAR(20),
	PaymentRecordId VARCHAR(10),
	FOREIGN KEY (EmployeeNumber) REFERENCES ShopStaff(EmployeeNumber) 
ON UPDATE CASCADE ON DELETE CASCADE,
	FOREIGN KEY (PaymentRecordId) REFERENCES 
StoreStaffPayment(PaymentRecordId)  ON UPDATE CASCADE ON DELETE CASCADE
	);


INSERT INTO ShopStaffShift VALUES (1,CONVERT(DATE, 'Oct 26 2019'), 
CONVERT(TIME, '9:00:00 AM'), CONVERT(DATE, 'Oct 26 2019'), CONVERT(TIME, 
'10:00:00 PM'), 101, '1 hr break', 1);

INSERT INTO ShopStaffShift VALUES (2,CONVERT(DATE, 'Oct 26 2019'), 
CONVERT(TIME, '10:00:00 AM'), CONVERT(DATE, 'Oct 26 2019'), CONVERT(TIME, 
'9:00:00 PM'), 102, '1 hr break', 2);

INSERT INTO ShopStaffShift VALUES (3,CONVERT(DATE, 'Oct 26 2019'), 
CONVERT(TIME, '11:00:00 AM'), CONVERT(DATE, 'Oct 26 2019'), CONVERT(TIME, 
'8:00:00 PM'), 103, '2 hr break', 3);


------------------------------------------------------------------------------------------------------------------------


--	Descriptive Attributes
CREATE TABLE IsAPartOf (
	OrderNo VARCHAR(10) NOT NULL,
	Code VARCHAR(20) NOT NULL,
	quantity INTEGER DEFAULT 1,
	price FLOAT,
	PRIMARY KEY (OrderNo, Code),
	FOREIGN KEY (OrderNo) REFERENCES IngredientOrder(OrderNo),
        FOREIGN KEY (Code) REFERENCES Ingredient(Code),
);

INSERT INTO IsAPartOf VALUES (200, 2, 3, 51.5);

INSERT INTO IsAPartOf VALUES (201, 1, 3, 30.75);
INSERT INTO IsAPartOf VALUES (201, 3, 4, 15.60);
INSERT INTO IsAPartOf VALUES (201, 4, 3, 10.45);

INSERT INTO IsAPartOf VALUES (202, 3, 3, 13);
INSERT INTO IsAPartOf VALUES (203, 4, 10, 34.85);

---------------------------------------------------------------------------------------------------------------

CREATE TABLE Supplies (
        SupplierNo VARCHAR(10) NOT NULL ,
        Code VARCHAR(20) NOT NULL ,
        PriceOfIngredients FLOAT,
        QuantityOfIngredients INTEGER DEFAULT 1,
        PRIMARY KEY (SupplierNo, Code),
        FOREIGN Key (SupplierNo) references Supplier (SupplierNo) ON UPDATE
CASCADE ON DELETE CASCADE,
        FOREIGN Key (Code) references Ingredient (Code) ON UPDATE CASCADE ON
DELETE CASCADE
);
INSERT INTO Supplies VALUES (007, 1, 30, 12);
INSERT INTO Supplies VALUES (70, 2, 60, 10);
INSERT INTO Supplies VALUES (21, 3, 25, 10);
INSERT INTO Supplies VALUES (21, 4, 27.5, 10);

---------------------------------------------------------------------------------------------------------------

CREATE TABLE includes (
	Code VARCHAR(20) NOT NULL  ,
	ItemCode VARCHAR(20) NOT NULL ,
	Quantity INTEGER NOT NULL DEFAULT 1,
	PRIMARY KEY (Code, ItemCode),
	FOREIGN Key (Code) references Ingredient (Code) ON UPDATE CASCADE ON DELETE CASCADE,
	FOREIGN Key (ItemCode) references MenuItems(ItemCode) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO includes VALUES (4, 001, 3);
INSERT INTO includes VALUES (3, 001, 6);
INSERT INTO includes VALUES (1, 001, 2);

INSERT INTO includes VALUES (1, 002, 7);

INSERT INTO includes VALUES (4, 003, 12);

INSERT INTO includes VALUES (2, 004, 7);
INSERT INTO includes VALUES (3, 004, 4);
INSERT INTO includes VALUES (1, 004, 3);

-------------------------------------------------------------------------------------------------------------

CREATE TABLE HasWalkIn(
	OrderNo CHAR(10) NOT NULL ,
	ItemCode VARCHAR(20) NOT NULL ,
	Quantity INTEGER DEFAULT 1,
	Price FLOAT,
	PRIMARY KEY (OrderNo , ItemCode),
	FOREIGN Key (OrderNo) references WalkInOrder (OrderNo) ON UPDATE CASCADE ON DELETE CASCADE,
	FOREIGN Key (ItemCode) references MenuItems (ItemCode) ON UPDATE CASCADE ON DELETE CASCADE
```

```
);

INSERT INTO HasWalkIn VALUES (60, 001, 5, 27);
INSERT INTO HasWalkIn VALUES (60, 004, 7, 67.84);

INSERT INTO HasWalkIn VALUES (61, 004, 10, 104.5);

INSERT INTO HasWalkIn VALUES (62, 002, 7, 92.4);

-----------------------------------------------------------------------------------------------------------

CREATE TABLE HasPickUp(
	OrderNo CHAR(10) NOT NULL ,
	ItemCode VARCHAR(20) NOT NULL ,
	Quantity INTEGER DEFAULT 1,
	Price FLOAT,
	PRIMARY KEY (OrderNo , ItemCode),
	FOREIGN Key (OrderNo) references PickupOrder (OrderNo)  ON UPDATE
CASCADE ON DELETE CASCADE,
	FOREIGN Key (ItemCode) references MenuItems (ItemCode) ON UPDATE
CASCADE ON DELETE CASCADE
);

INSERT INTO HasPickUp VALUES (170, 003, 3, 21.3);

INSERT INTO HasPickUp VALUES (171, 001, 4, 26.7);

INSERT INTO HasPickUp VALUES (172, 004, 6, 45.9);

-----------------------------------------------------------------------------------------------------------

CREATE TABLE HasDelivery(
	OrderNo CHAR(10) NOT NULL ,
        ItemCode VARCHAR(20) NOT NULL ,
        Quantity INTEGER DEFAULT 1,
        Price FLOAT,
        PRIMARY KEY (OrderNo , ItemCode),
        FOREIGN Key (OrderNo) references DeliveryOrder (OrderNo) ON UPDATE
CASCADE ON DELETE CASCADE,
        FOREIGN Key (ItemCode) references MenuItems (ItemCode) ON UPDATE
CASCADE ON DELETE CASCADE
);



INSERT INTO HasDelivery VALUES (1, 004, 12, 135.5);


INSERT INTO HasDelivery VALUES (2, 002, 6, 33);
INSERT INTO HasDelivery VALUES (2, 003, 7, 27);


INSERT INTO HasDelivery VALUES (3, 004, 12, 122.4);

------------------------------------------------------------------------------------------------------------------

-- Assignment Questions:

-- Q.1 For an in-office staff with id number xxx, print his/her 1stname, lname, AND
hourly payment rate.
SELECT EmployeeNumber, FirstName, LastName, PaymentRate
FROM ShopStaff
WHERE EmployeeNumber = 102;

-- -- Q.2 List all the shift details of a delivery staff with first name xxx AND last
name ttt BETWEEN date yyy AND zzz.
SELECT s.*
FROM DeliveryStaffShift s, DriverStaff d
WHERE s.EmployeeNumber = d.EmployeeNumber AND
```

```
s.EmployeeNumber = (SELECT EmployeeNumber FROM DriverStaff WHERE FirstName =
'Codey' AND LastName = 'Price')

AND s.StartDate BETWEEN CONVERT(DATE, 'Oct 1 2019') AND CONVERT(DATE, 'Oct
31 2019') AND

s.EndDate BETWEEN CONVERT(DATE, 'Oct 1 2019') AND CONVERT(DATE, 'Oct 31
2019');


-- -- Q.3 List all the order details of the orders that are made by a walk-in customer
with  first name xxx AND last name ttt BETWEEN date yyy AND zzz.

SELECT w.*

FROM WalkInOrder w, Customer c

WHERE w.CusID = c.CusID AND w.CusID = (SELECT CusID FROM Customer WHERE
fName = 'Ryan' AND lName ='Goff') AND

w.OrderDateTime BETWEEN CONVERT(DATE, 'Oct 1 2019') AND CONVERT(DATE, 'Oct
31 2019');


-- -- Q.4 Print the salary paid to a delivery staff with first name xxx AND last name
ttt  in current  month.

-- -- Note the current month is the current month that is decided by the system.


SELECT d.EmployeeNumber, SUM(GrossPayment) AS GrossPayment,
SUM(p.TotalAmountPaid) AS TotalAmountPaid

FROM DeliveryStaffShift s, DriverStaff d, DriverStaffPayment p

WHERE d.EmployeeNumber = s.EmployeeNumber

        AND s.PaymentRecordId = p.PaymentRecordId

        AND s.EmployeeNumber = (SELECT EmployeeNumber FROM DriverStaff WHERE
FirstName = 'Codey' AND LastName = 'Price')

        AND DATEPART(MONTH, p.PaymentDate) = MONTH(GETDATE())

GROUP BY d.EmployeeNumber


-- -- Q.5 List the name of the menu item that is mostly ordered in current year


CREATE TABLE #MenuOrderTimes

(

        Name CHAR(20),
        Quantity INTEGER
);

INSERT into #MenuOrderTimes
SELECT DISTINCT s.Name, Quantity
FROM HasPickUp h, MenuItems s, PickUpOrder p
WHERE h.ItemCode = s.ItemCode AND DATEPART(YEAR, (p.OrderDateTime)) =
YEAR(getdate());

INSERT into #MenuOrderTimes
SELECT DISTINCT s.Name, Quantity
FROM HasWalkIn h, MenuItems s, WalkInOrder w
WHERE h.ItemCode = s.ItemCode AND DATEPART(YEAR, (w.OrderDateTime)) =
YEAR(getdate());

INSERT into #MenuOrderTimes
SELECT DISTINCT s.Name, Quantity
FROM HasDelivery h, MenuItems s, DeliveryOrder d
WHERE h.ItemCode = s.ItemCode AND DATEPART(YEAR, (d.OrderDateTime)) =
YEAR(getdate());

CREATE TABLE #TotalMenuItemsOrder
(
        Name CHAR(20),
        Quantity INTEGER
);
INSERT into #TotalMenuItemsOrder SELECT Name, SUM(Quantity) FROM
#MenuOrderTimes GROUP BY Name;

SELECT Name, Quantity FROM #TotalMenuItemsOrder WHERE Quantity >= ALL (SELECT
Quantity FROM #TotalMenuItemsOrder) GROUP BY Name, Quantity;

DROP TABLE #MenuOrderTimes;
DROP TABLE #TotalMenuItemsOrder;
```

```sql
-- -- Q.6  List the name(s) of the  ingredient(s) that was/were supplied by the supplier with
supplier ID xxx on date yyy.


SELECT Distinct i.Name

FROM IngredientOrder io, IsAPartOf p, Ingredient i

WHERE io.SupplierNo = 70 AND io.OrderNo = p.OrderNo AND i.Code = p.Code AND
io.DateReviecedOrder = CONVERT(date, 'Oct 26 2019');
```