INFORMATION AND DATA

ECM1420

Database Design and Implementation Exercise

Student Number: 700073497

Table of Contents

1. Introduction:

In this coursework, I will be coding the solution of the assigned exercises in SQL. In tables, the columns have suitable data types, auto increment where necessary, NULL/NOT NULL has been set appropriately and foreign keys created, respectively. I created view, provided/run the solution for all assigned stored procedures and generated SQL database diagram. I believe, it correctly solves the problem and displays the output for the given tasks. The meaningful naming conventions, extra spaces and indentations is used to improve the readability. There are no errors in the SQL script and the solution code for each task with output/result is mentioned as below.

2. SQL code from action point 2 to create the following database tables ONLY, including primary keys; StockCategory, StockItem, Customer, SalesOrder, SalesOrderLine, SupplierStockItem.

StockCategory

```
USE [NymptonFoodHub]
GO

/****** Object:  Table [dbo].[StockCategory]    Script Date: 21/03/2021 04:14:56 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[StockCategory](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[Category] [nvarchar](200) NULL,
	[DisplayOrder] [int] NULL,
 CONSTRAINT [PK_StockCategory] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

StockItem

```
USE [NymptonFoodHub]
GO

/****** Object:  Table [dbo].[StockItem]    Script Date: 21/03/2021 04:23:05 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[StockItem](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[ItemName] [nvarchar](200) NULL,
	[ItemUnit] [nvarchar](200) NULL,
	[ItemPrice] [money] NULL,
	[Available] [nvarchar](50) NULL,
	[ItemCategory] [int] NULL,
	[ItemAdditionalInfo] [nvarchar](200) NULL,
 CONSTRAINT [PK_StockItem] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Customer

```
USE [NymptonFoodHub]
GO

/****** Object:  Table [dbo].[Customer]    Script Date: 21/03/2021 04:25:29 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Customer](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[Contact_id] [int] NOT NULL,
	[Add_Info] [nvarchar](100) NULL,
	[Delivery_info] [nvarchar](250) NULL,
 CONSTRAINT [PK_Customer] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SalesOrder

```
USE [NymptonFoodHub]
GO

/****** Object:  Table [dbo].[SalesOrder]    Script Date: 21/03/2021 04:26:13 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[SalesOrder](
	[Order_id] [int] IDENTITY(1,1) NOT NULL,
	[Customer_id] [int] NOT NULL,
	[OrderDate] [date] NULL,
	[Picked] [nvarchar](50) NULL,
	[Delivered] [nvarchar](50) NULL,
	[Invoiced] [nvarchar](50) NULL,
	[Paid] [nvarchar](50) NULL,
	[Complete] [nvarchar](50) NULL,
	[GoodsCost] [money] NULL,
	[ExtrasText] [nvarchar](250) NULL,
	[ExtrasCost] [money] NULL,
	[DeliveryCost] [money] NULL,
	[TotalCost] [money] NULL,
 CONSTRAINT [PK_SalesOrder] PRIMARY KEY CLUSTERED
(
	[Order_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SalesOrderLine

```
USE [NymptonFoodHub]
GO

/****** Object:  Table [dbo].[SalesOrderLine]    Script Date: 21/03/2021 04:28:14 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[SalesOrderLine](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[Order_id] [int] NOT NULL,
	[Item_id] [int] NOT NULL,
	[ItemQty] [int] NULL,
	[ItemName] [nvarchar](200) NULL,
	[ItemUnit] [nvarchar](100) NULL,
	[ItemPrice] [money] NULL,
	[ItemCost] [money] NULL,
 CONSTRAINT [PK_SalesOrderLine] PRIMARY KEY CLUSTERED 
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, 
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

SupplierStockItem

```
USE [NymptonFoodHub]
GO

/****** Object:  Table [dbo].[SupplierStockItem]    Script Date: 21/03/2021 04:29:12 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[SupplierStockItem](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[Supplier_id] [int] NOT NULL,
	[StockItem_id] [int] NOT NULL,
	[Add_Info] [nvarchar](200) NULL,
 CONSTRAINT [PK_SupplierStockItem] PRIMARY KEY CLUSTERED 
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, 
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

3. The SQL code that I used in action point 4 to create the Foreign Key constraints in the SupplierStockItem and CustomerPayment tables only.

SupplierStockItem

```
USE [NymptonFoodHub]
GO

ALTER TABLE [dbo].[SupplierStockItem]  WITH CHECK ADD  CONSTRAINT [FK_SupplierStockItem_StockItem]
FOREIGN KEY([StockItem_id])
REFERENCES [dbo].[StockItem] ([Id])
GO

ALTER TABLE [dbo].[SupplierStockItem] CHECK CONSTRAINT [FK_SupplierStockItem_StockItem]
GO

ALTER TABLE [dbo].[SupplierStockItem]  WITH CHECK ADD  CONSTRAINT [FK_SupplierStockItem_Supplier]
FOREIGN KEY([Supplier_id])
REFERENCES [dbo].[Supplier] ([Id])
GO

ALTER TABLE [dbo].[SupplierStockItem] CHECK CONSTRAINT [FK_SupplierStockItem_Supplier]
GO
```

CustomerPayment

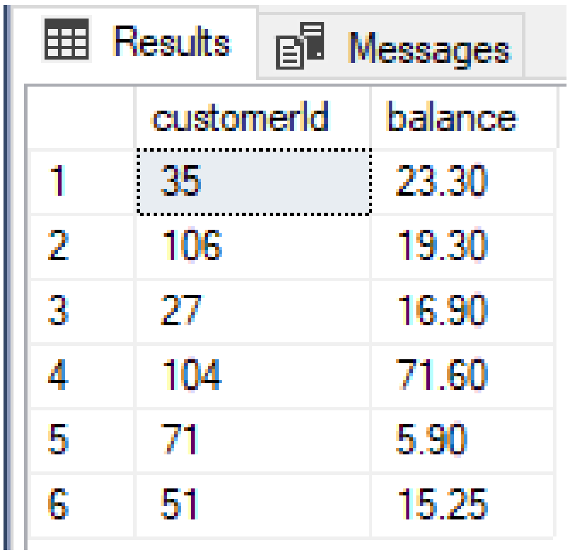
```
USE [NymptonFoodHub]
GO

ALTER TABLE [dbo].[CustomerPayment]  WITH CHECK ADD  CONSTRAINT [FK_CustomerPayment_Customer]
FOREIGN KEY([Customer_id])
REFERENCES [dbo].[Customer] ([Id])
GO

ALTER TABLE [dbo].[CustomerPayment] CHECK CONSTRAINT [FK_CustomerPayment_Customer]
GO

ALTER TABLE [dbo].[CustomerPayment]  WITH CHECK ADD  CONSTRAINT [FK_CustomerPayment_SalesInvoice]
FOREIGN KEY([Invoice_id])
REFERENCES [dbo].[SalesInvoice] ([Id])
GO

ALTER TABLE [dbo].[CustomerPayment] CHECK CONSTRAINT [FK_CustomerPayment_SalesInvoice]
GO

ALTER TABLE [dbo].[CustomerPayment]  WITH CHECK ADD  CONSTRAINT [FK_CustomerPayment_SalesOrder]
FOREIGN KEY([Order_id])
REFERENCES [dbo].[SalesOrder] ([Order_id])
GO

ALTER TABLE [dbo].[CustomerPayment] CHECK CONSTRAINT [FK_CustomerPayment_SalesOrder]
GO
```

4. SQL script that I used to create the View named CustomerDebt.

CustomerDebt SQL script

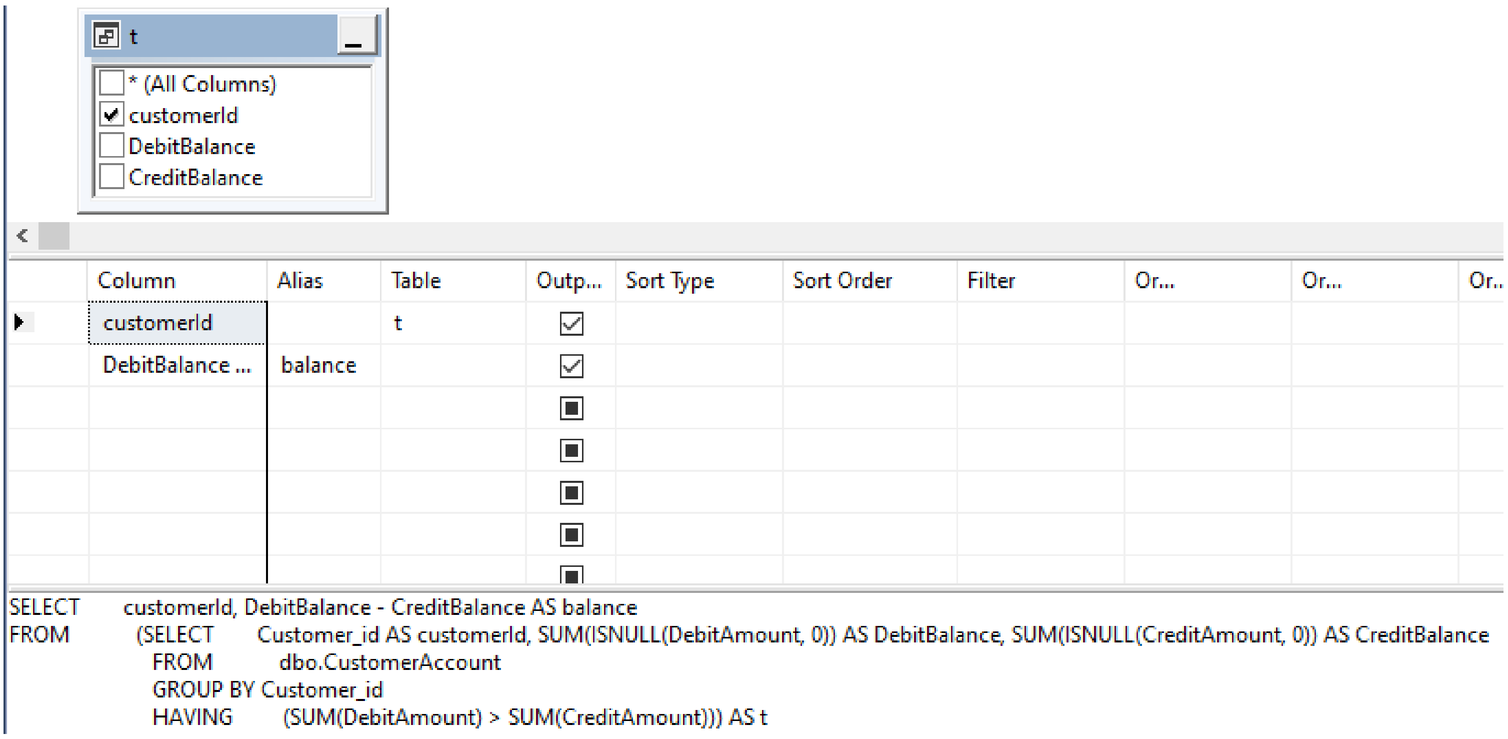
```
USE [NymptonFoodHub]
GO

/****** Object:  View [dbo].[CustomerDebt]    Script Date: 21/03/2021 04:48:10 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE view [dbo].[CustomerDebt] as
Select customerId, (DebitBalance - CreditBalance) as balance from (
Select
Customer_id as customerId,
SUM(ISNULL(DebitAmount,0)) as DebitBalance,
SUM(ISNULL(CreditAmount,0)) as CreditBalance
From CustomerAccount
Group By Customer_id
Having SUM(DebitAmount)>SUM(CreditAmount)
) as t
GO
```

CustomerDebt Output:



	customerId	balance
1	35	23.30
2	106	19.30
3	27	16.90
4	104	71.60
5	71	5.90
6	51	15.25

CustomerDebt Design:



Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
customerId		t	✓						
DebitBalance ...	balance		✓						

```
SELECT      customerId, DebitBalance - CreditBalance AS balance
FROM          (SELECT      Customer_id AS customerId, SUM(ISNULL(DebitAmount, 0)) AS DebitBalance, SUM(ISNULL(CreditAmount, 0)) AS CreditBalance
                  FROM          dbo.CustomerAccount
                  GROUP BY Customer_id
                  HAVING       (SUM(DebitAmount) > SUM(CreditAmount))) AS t
```

5. SQL five stored procedures from action point 6 with output together with the SQL code for each report from action point 7.

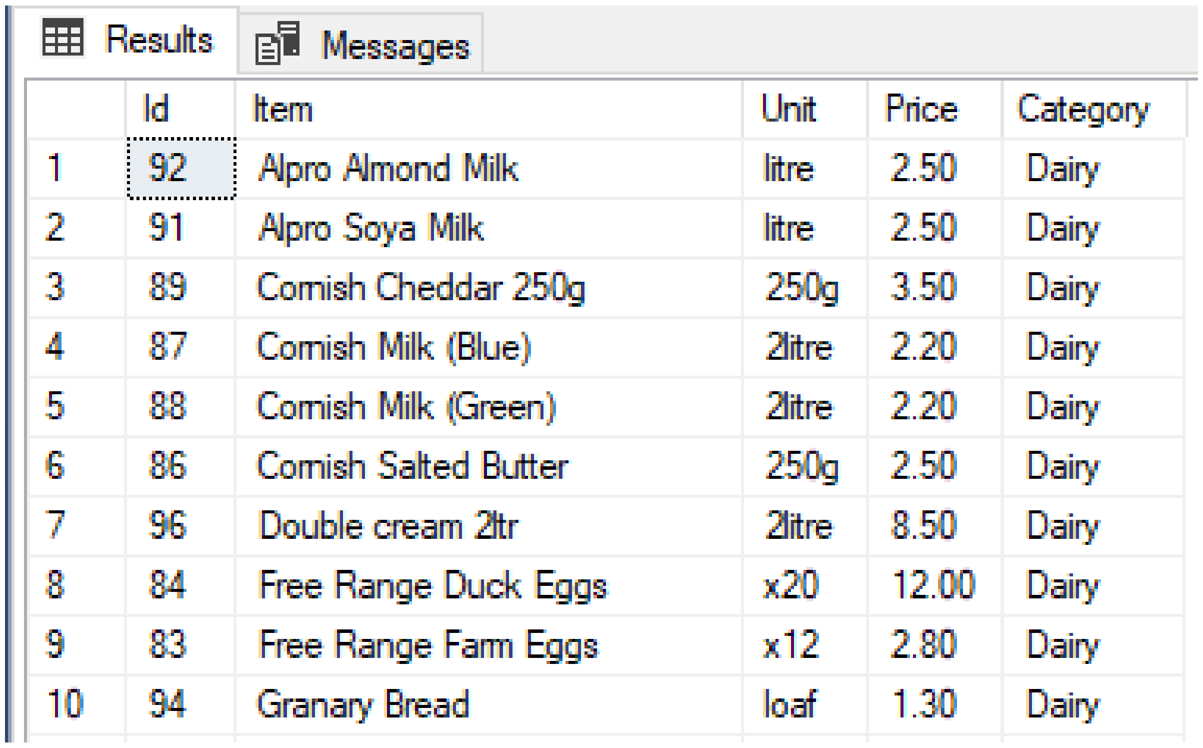
The SQL script/code for the five stored procedures is as following:

5.1 Price List by Category/Item

SQL script

```
USE [NymptonFoodHub]
GO
/****** Object:  StoredProcedure [dbo].[SpGetCategoriesPriceByItem_1]    Script Date: 21/03/2021
05:32:55 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[SpGetCategoriesPriceByItem_1]

AS
BEGIN

	SET NOCOUNT ON;
	Select
	SI.Id,
	SI.ItemName as Item,
	SI.ItemUnit as Unit,
	SI.ItemPrice as Price,
	SC.Category

	from StockItem SI LEFT JOIN StockCategory SC on SI.ItemCategory = SC.Id

	order by SC.Category ASC, SI.ItemName
END

--EXEC [SpGetCategoriesPriceByItem_1];
GO
```

Output



Results | Messages

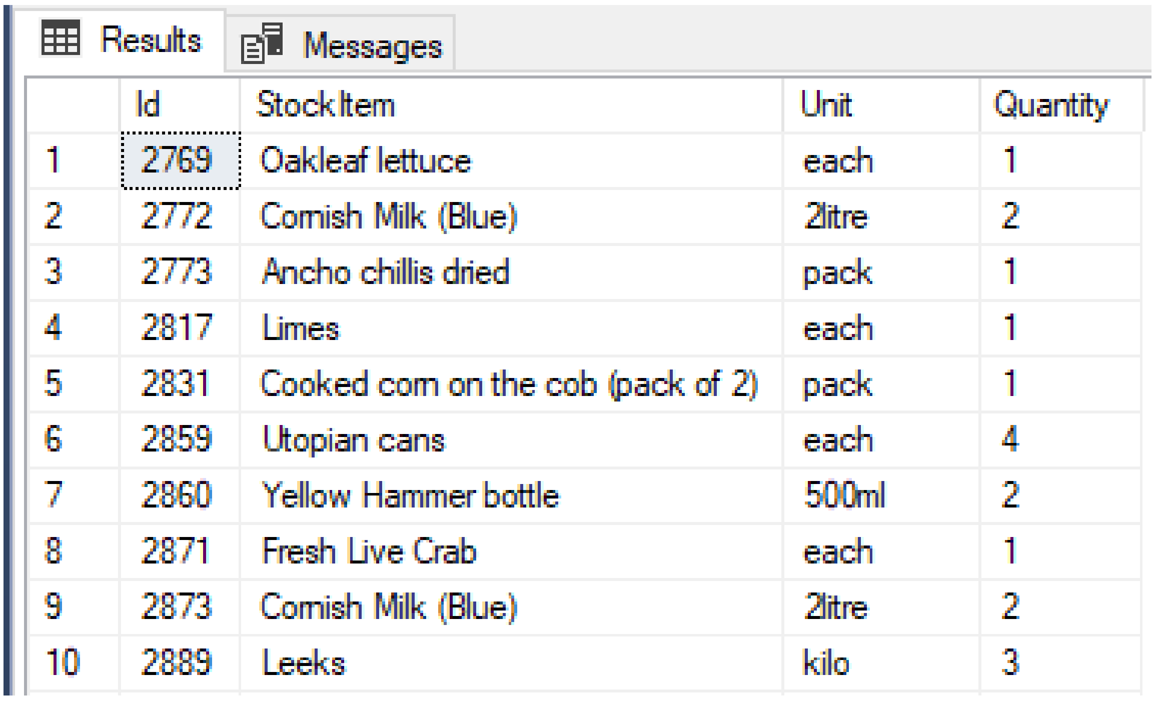
	Id	Item	Unit	Price	Category
1	92	Alpro Almond Milk	litre	2.50	Dairy
2	91	Alpro Soya Milk	litre	2.50	Dairy
3	89	Cornish Cheddar 250g	250g	3.50	Dairy
4	87	Cornish Milk (Blue)	2litre	2.20	Dairy
5	88	Cornish Milk (Green)	2litre	2.20	Dairy
6	86	Cornish Salted Butter	250g	2.50	Dairy
7	96	Double cream 2ltr	2litre	8.50	Dairy
8	84	Free Range Duck Eggs	x20	12.00	Dairy
9	83	Free Range Farm Eggs	x12	2.80	Dairy
10	94	Granary Bread	loaf	1.30	Dairy

5.2 Picking List

SQL script

```
USE [NymptonFoodHub]
GO
/****** Object:  StoredProcedure [dbo].[spGetPickingItemsByQuantity_2]    Script Date: 21/03/2021
05:35:07 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[spGetPickingItemsByQuantity_2]

AS
BEGIN
	SET NOCOUNT ON;
	SELECT
	SL.Id,
	SL.ItemName as StockItem,
	SL.ItemUnit as Unit,
	SL.ItemQty as Quantity
	from  SalesOrderLine SL INNER JOIN SalesOrder SO on SL.Order_id = SO.Order_id
	where SO.Picked='N'
END

--EXEC [spGetPickingItemsByQuantity_2];
GO
```

Output



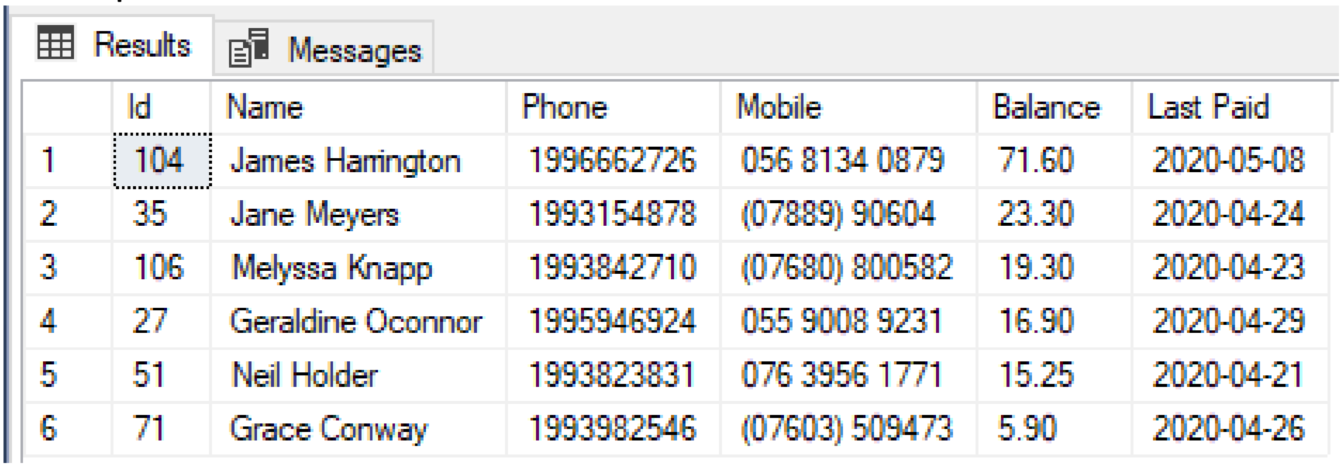
	Id	StockItem	Unit	Quantity
1	2769	Oakleaf lettuce	each	1
2	2772	Cornish Milk (Blue)	2litre	2
3	2773	Ancho chillis dried	pack	1
4	2817	Limes	each	1
5	2831	Cooked corn on the cob (pack of 2)	pack	1
6	2859	Utopian cans	each	4
7	2860	Yellow Hammer bottle	500ml	2
8	2871	Fresh Live Crab	each	1
9	2873	Cornish Milk (Blue)	2litre	2
10	2889	Leeks	kilo	3

5.3 Customer Outstanding Balances

SQL script

```
USE [NymptonFoodHub]
GO
/****** Object:  StoredProcedure [dbo].[SpGetCustomerOutstandingBalance_3]    Script Date: 
21/03/2021 05:41:13 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[SpGetCustomerOutstandingBalance_3]

AS
BEGIN

SELECT
CustomerDebt.customerId Id,
Contact.FirstName + ' ' + Contact.Surname [Name],
Contact.PhoneLandline Phone,
Contact.PhoneMobile Mobile,
CustomerDebt.balance Balance,
(Select MAX(ReceiptDate) from CustomerPayment CP Where CP.Customer_id = Customer.Id) [Last Paid]

FROM
CustomerDebt INNER JOIN
Customer ON CustomerDebt.customerId = Customer.Id INNER JOIN
Contact ON Customer.Contact_id = Contact.Id
Order by balance desc

END

--EXEC [SpGetCustomerOutstandingBalance_3];
GO
```

Output


Results Messages

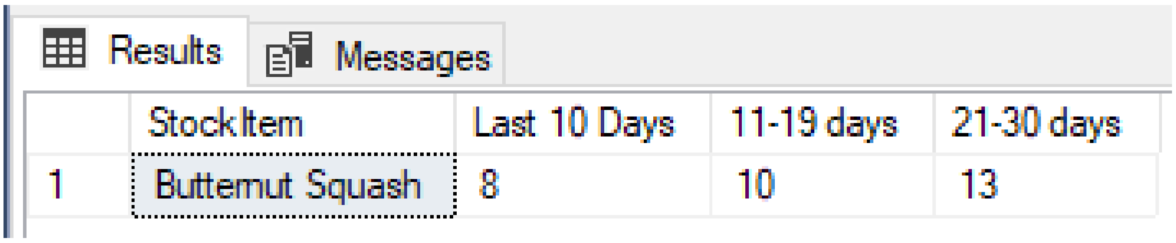
	Id	Name	Phone	Mobile	Balance	Last Paid
1	104	James Harrington	1996662726	056 8134 0879	71.60	2020-05-08
2	35	Jane Meyers	1993154878	(07889) 90604	23.30	2020-04-24
3	106	Melyssa Knapp	1993842710	(07680) 800582	19.30	2020-04-23
4	27	Geraldine Oconnor	1995946924	055 9008 9231	16.90	2020-04-29
5	51	Neil Holder	1993823831	076 3956 1771	15.25	2020-04-21
6	71	Grace Conway	1993982546	(07603) 509473	5.90	2020-04-26

5.4 Recent Demand for Stock Item

SQL script

```
USE [NymptonFoodHub]
GO
/****** Object:  StoredProcedure [dbo].[SpRecentDemandforStockItem_4]    Script Date: 21/03/2021
05:49:03 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- exec [SpRecentDemandforStockItem_4] '2020-05-01', 'Butternut Squash'

CREATE PROCEDURE [dbo].[SpRecentDemandforStockItem_4]

@OrderDate DateTime,
@Name nvarchar(250)

AS
BEGIN

SET NOCOUNT ON;

select StockItem ,sum(Last10) as [Last 10 Days],Sum(Last20) as [11-19 days],Sum(Last30) as [21-30
days] from (
SELECT
SL.ItemName as StockItem,
(select count(Sl.ItemQty) WHERE DATEDIFF(day,SO.OrderDate,@OrderDate) between 0 and 10) as Last10,
(select count(Sl.ItemQty) WHERE DATEDIFF(day,SO.OrderDate,@OrderDate) between 11 and 19) as Last20,
(select count(Sl.ItemQty) WHERE DATEDIFF(day,SO.OrderDate,@OrderDate) between 21 and 30) as Last30
FROM  SalesOrderLine SL INNER JOIN SalesOrder SO on  SL.Order_id = SO.Order_id
where Sl.ItemName=@Name
Group by SL.ItemName,SO.OrderDate
) as t Group By StockItem
END
GO
```

Output



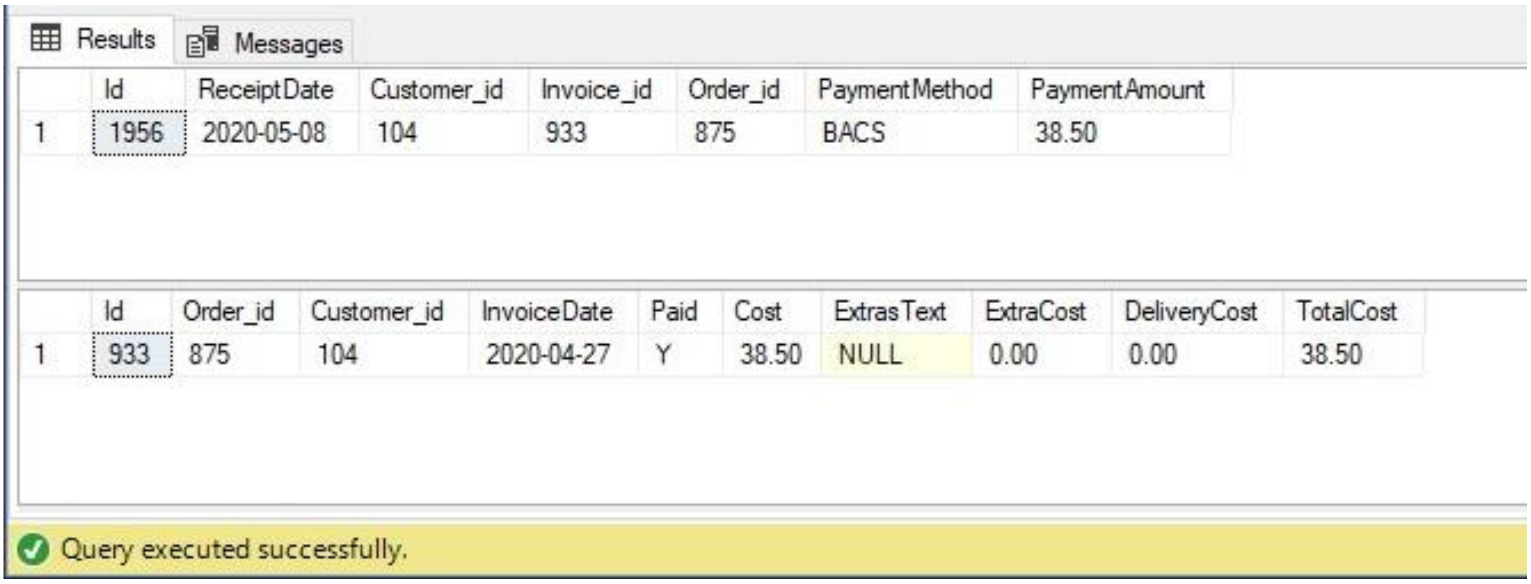
	StockItem	Last 10 Days	11-19 days	21-30 days
1	Butternut Squash	8	10	13

5.5 Receive Full Payment on Invoice

SQL script

```
USE [NymptonFoodHub]
GO
/****** Object:  StoredProcedure [dbo].[SpReceiveFullPaymentOnInvoice_5]    Script Date: 21/03/2021
05:54:52 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

 -- exec [SpReceiveFullPaymentOnInvoice_5] 933,'2020-05-08'

CREATE PROCEDURE [dbo].[SpReceiveFullPaymentOnInvoice_5]

       @InvoiceId int,
       @PaymentDate Date

AS
declare @NewId as table(NewId int)
IF EXISTS (select id from SalesInvoice where id = @InvoiceId and paid = 'N')
BEGIN
INSERT INTO CustomerPayment (ReceiptDate, Customer_id, Invoice_id, Order_id, PaymentMethod,
PaymentAmount)
OUTPUT INSERTED.id INTO @NewId

SELECT @PaymentDate, Customer_id, @InvoiceId, Order_id, 'BACS', inv.totalcost from SalesInvoice as
inv
WHERE inv.id=@InvoiceId
SELECT * from CustomerPayment Where CustomerPayment.id = (SELECT NewId FROM @NewID)

UPDATE SalesInvoice SET paid ='Y' WHERE id = @InvoiceId

END

SELECT * from SalesInvoice where SalesInvoice.id = @InvoiceId
GO
```

Output:



	Id	ReceiptDate	Customer_id	Invoice_id	Order_id	PaymentMethod	PaymentAmount
1	1956	2020-05-08	104	933	875	BACS	38.50

	Id	Order_id	Customer_id	InvoiceDate	Paid	Cost	ExtrasText	ExtraCost	DeliveryCost	TotalCost
1	933	875	104	2020-04-27	Y	38.50	NULL	0.00	0.00	38.50

Query executed successfully.

6. Explanation of revisions required to procedure to receive full payment on an invoice.

If the customer payment record creation failed, then we should not set Paid = 'Y' in sales invoice record.

To resolve this issue, we will use try and catch inside the SQL commitment control in which if one of the query fails then other query should not work and the roll back process will happen.

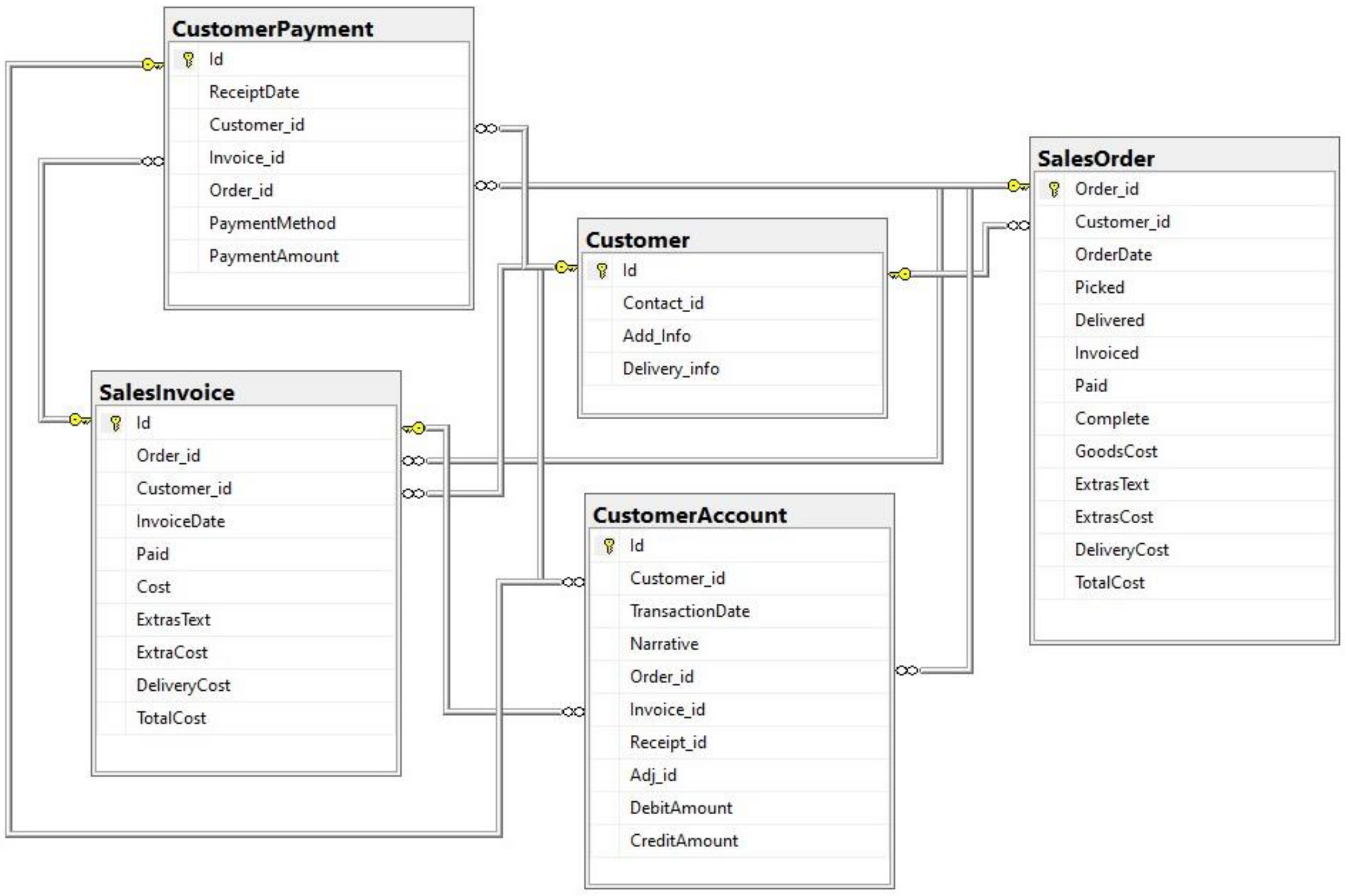
This can be handled by using try and catch statements of T-SQL which supports this case, if a single query fails then it goes directly into the catch block. Further, we have used commit and roll back inside the try catch block to handle the error as mentioned in the question. The general syntax is as following:

```
      BEGIN TRY
           BEGIN TRAN
           { sql_statement | statement_block }
           if the transaction is successful, we have to commit the
transaction here.
           COMMIT TRAN
      END TRY
      BEGIN CATCH
           [ { sql_statement | statement_block } ]
           if the transaction is unsuccessful, we have to perform
rollback
           ROLLBACK TRAN
      END CATCH
      [ ; ]
```

The major part of the queries will be written in try section. If all the queries works properly then there is an end a statement called 'commit' which will be executed.

Else, If the catch block catches an error, then the statement 'ROLLBACK' will be executed in which roll back process will happen.

7. Database diagram from action point 9, using the SSMS database diagram tool to show tables and their columns, relationships between the tables and their cardinality.


CustomerPayment
Id
ReceiptDate
Customer_id
Invoice_id
Order_id
PaymentMethod
PaymentAmount
Customer
Id
Contact_id
Add_Info
Delivery_info
SalesOrder
Order_id
Customer_id
OrderDate
Picked
Delivered
Invoiced
Paid
Complete
GoodsCost
ExtrasText
ExtrasCost
DeliveryCost
TotalCost
SalesInvoice
Id
Order_id
Customer_id
InvoiceDate
Paid
Cost
ExtrasText
ExtraCost
DeliveryCost
TotalCost
CustomerAccount
Id
Customer_id
TransactionDate
Narrative
Order_id
Invoice_id
Receipt_id
Adj_id
DebitAmount
CreditAmount