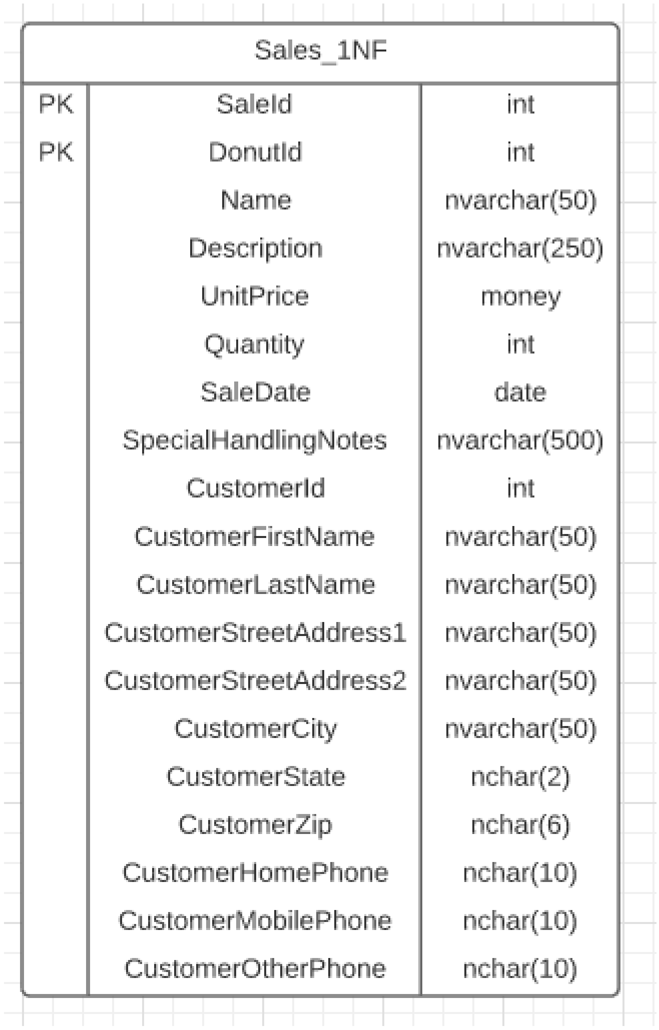
Duncan Nisbett
Student ID: #000856539
C170: Data Management - Applications

Data Management Project

Part A Normalized Model

1a First Normal Form

Table Design



Sales_1NF		
PK	SaleId	int
PK	DonutId	int
	Name	nvarchar(50)
	Description	nvarchar(250)
	UnitPrice	money
	Quantity	int
	SaleDate	date
	SpecialHandlingNotes	nvarchar(500)
	CustomerId	int
	CustomerFirstName	nvarchar(50)
	CustomerLastName	nvarchar(50)
	CustomerStreetAddress1	nvarchar(50)
	CustomerStreetAddress2	nvarchar(50)
	CustomerCity	nvarchar(50)
	CustomerState	nchar(2)
	CustomerZip	nchar(6)
	CustomerHomePhone	nchar(10)
	CustomerMobilePhone	nchar(10)
	CustomerOtherPhone	nchar(10)

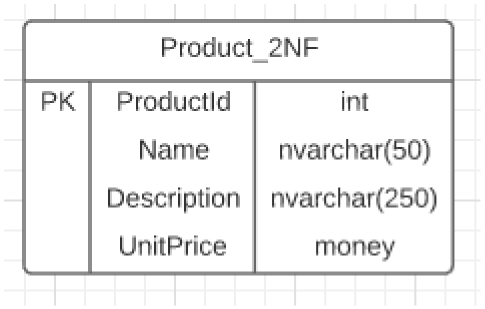
```
CREATE TABLE [dbo].[Sales_1NF]
(
	[SaleId] INT NOT NULL IDENTITY(1,1),
	[DonutId] INT NOT NULL,
	[Name] NVARCHAR(50) NOT NULL,
	[Description] NVARCHAR(250) NULL,
	[UnitPrice] MONEY NULL,
	[Quantity] INT NOT NULL,
	[SaleDate] DATE NOT NULL,
	[SpecialHandlingNotes] NVARCHAR(500) NULL,
	[CustomerId] INT NULL,
	[CustomerFirstName] NVARCHAR(50) NULL,
	[CustomerLastName] NVARCHAR(50) NULL,
	[CustomerStreetAddress1] NVARCHAR(50) NULL,
	[CustomerStreetAddress2] NVARCHAR(50) NULL,
	[CustomerCity] NVARCHAR(50) NULL,
	[CustomerState] NCHAR(2) NULL,
	[CustomerZip] NCHAR(6) NULL,
	[CustomerHomePhone] NCHAR(10) NULL,
	[CustomerMobilePhone] NCHAR(10) NULL,
	[CustomerOtherPhone] NCHAR(10) NULL,
	CONSTRAINT [PK_Sales_1NF] PRIMARY KEY ([SaleId],[DonutId])
)
```

Reasoning

I took the Sales form sheet and reviewed the data to break out each individual artifact. The table has been broken up based on the requirements and the unique data points found within the form. From there I used a standard naming convention to give each data point a self describing name like, CustomerFirstName, to make a clear designation on the type of value one could find in the column. Each data point was also examined to determine what type of data it best represented. A whole number such as id or count column was assigned as an integer, any short text string stored as nchar, longer text strings stored as nvarchar, and then money for the unit price. The Primary Key was derived as being the SaleId and DonutId. A composite key with those 2 data point enforces uniqueness for each record.

1b Second Normal Form

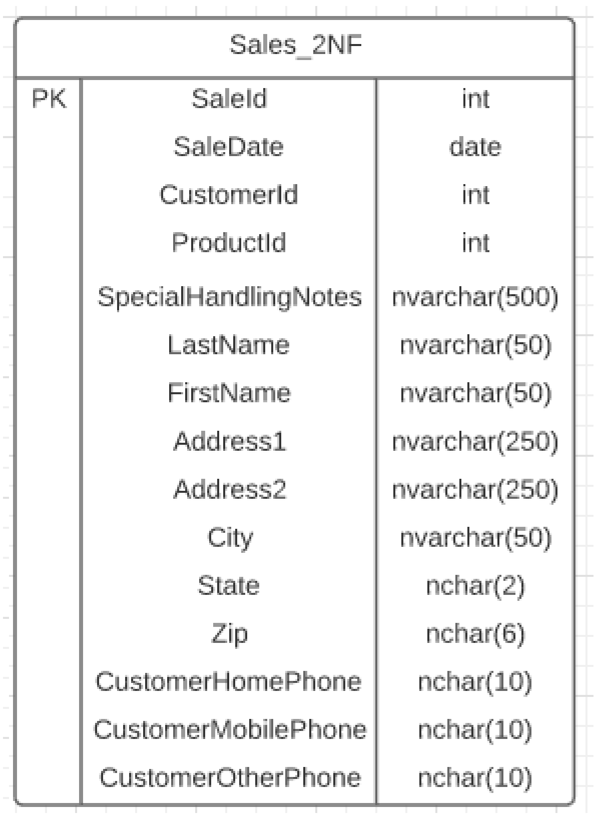
Table Design



Product_2NF		
PK	ProductId	int
	Name	nvarchar(50)
	Description	nvarchar(250)
	UnitPrice	money

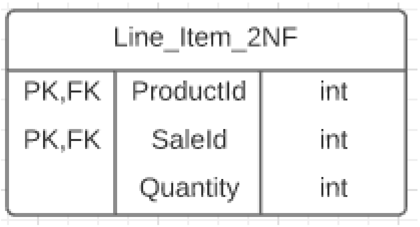
```
CREATE TABLE [dbo].[Product_2NF]
(
	[ProductId] INT NOT NULL IDENTITY(1,1),
	[Name] NVARCHAR(50) NOT NULL,
	[Description] NVARCHAR(250) NOT NULL,
	[UnitPrice] MONEY NOT NULL,
	CONSTRAINT [PK_Product_2NF] PRIMARY KEY (ProductId)
)
```

Duncan Nisbett
Student ID: #000856539
C170: Data Management - Applications



Sales_2NF		
PK	SaleId	int
	SaleDate	date
	CustomerId	int
	ProductId	int
	SpecialHandlingNotes	nvarchar(500)
	LastName	nvarchar(50)
	FirstName	nvarchar(50)
	Address1	nvarchar(250)
	Address2	nvarchar(250)
	City	nvarchar(50)
	State	nchar(2)
	Zip	nchar(6)
	CustomerHomePhone	nchar(10)
	CustomerMobilePhone	nchar(10)
	CustomerOtherPhone	nchar(10)

```
CREATE TABLE [dbo].[Sales_2NF]
(
	[SaleId] INT NOT NULL IDENTITY(1,1),
	[SaleDate] DATE NOT NULL,
	[CustomerId] INT NOT NULL,
	[ProductId] INT NOT NULL,
	[SpecialHandlingNotes] NVARCHAR(500) NULL,
	[LastName] NVARCHAR(50) NOT NULL,
	[FirstName] NVARCHAR(50) NOT NULL,
	[Address1] NVARCHAR(250) NOT NULL,
	[Address2] NVARCHAR(250) NULL,
	[City] NVARCHAR(50) NOT NULL,
	[State] NCHAR(2) NOT NULL,
	[Zip] NCHAR(6) NOT NULL,
	[CustomerHomePhone] NCHAR(10) NULL,
	[CustomerMobilePhone] NCHAR(10) NULL,
	[CustomerOtherPhone] NCHAR(10) NULL,
	CONSTRAINT [PK_Sales_2NF] PRIMARY KEY ([SaleId]),
)
```



Line_Item_2NF		
PK,FK	ProductId	int
PK,FK	SaleId	int
	Quantity	int

```
CREATE TABLE [dbo].[Line_Item_2NF]
```

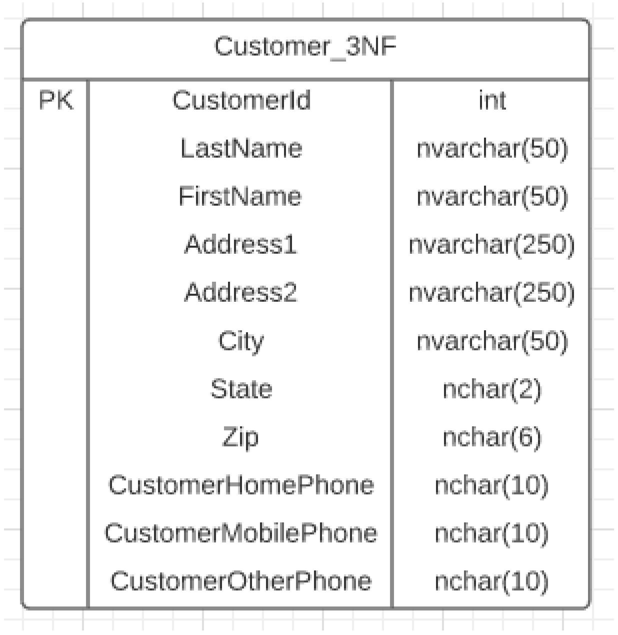
```
(
	[ProductId] INT NOT NULL,
	[SaleId] INT NOT NULL,
	[Quantity] INT NOT NULL,
	CONSTRAINT [FK_Line_Item_2NF_Sale] FOREIGN KEY ([SaleId]) REFERENCES
[Sales_2NF]([SaleId]),
	CONSTRAINT [FK_Line_Item_2NF_Product] FOREIGN KEY ([ProductId]) REFERENCES
[Product_2NF]([ProductId]),
	CONSTRAINT [PK_Line_Item_2NF] PRIMARY KEY ([SaleId], [ProductId])
)
```

Reasoning

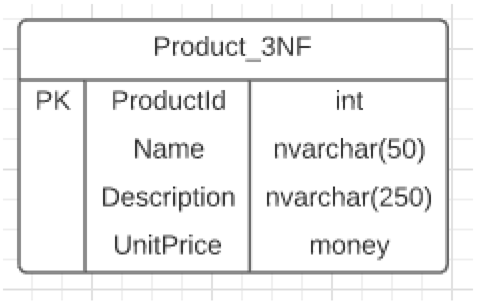
The sales data was broken out to 3 sections. Product data to store the information into each individual item that can be sold. The Sales table tracks information specific to each transaction along with the individual components of the sales data including the customer data. Finally a Line Item table was created to store the information for each item linked to a sale. 2 foreign keys are configured to enforce that a line item must be linked to a valid sale and to a valid product. As the 2 foreign keys also end up creating a unique record the same keys were configured as a composite primary key.

1c Third Normal Form

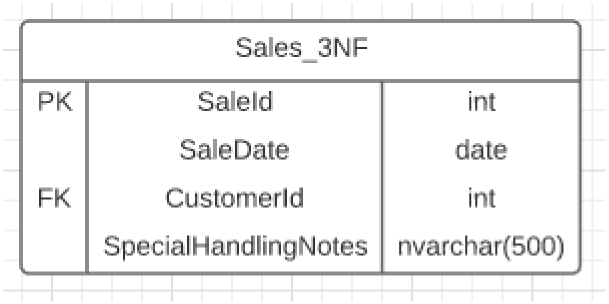
Table Design



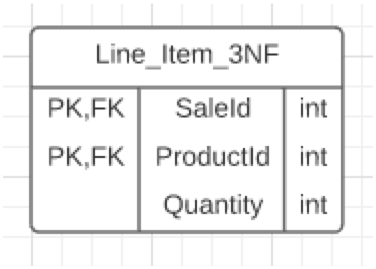
Customer_3NF		
PK	CustomerId	int
	LastName	nvarchar(50)
	FirstName	nvarchar(50)
	Address1	nvarchar(250)
	Address2	nvarchar(250)
	City	nvarchar(50)
	State	nchar(2)
	Zip	nchar(6)
	CustomerHomePhone	nchar(10)
	CustomerMobilePhone	nchar(10)
	CustomerOtherPhone	nchar(10)



Product_3NF		
PK	ProductId	int
	Name	nvarchar(50)
	Description	nvarchar(250)
	UnitPrice	money



Sales_3NF		
PK	SaleId	int
	SaleDate	date
FK	CustomerId	int
	SpecialHandlingNotes	nvarchar(500)



Line_Item_3NF		
PK,FK	SaleId	int
PK,FK	ProductId	int
	Quantity	int

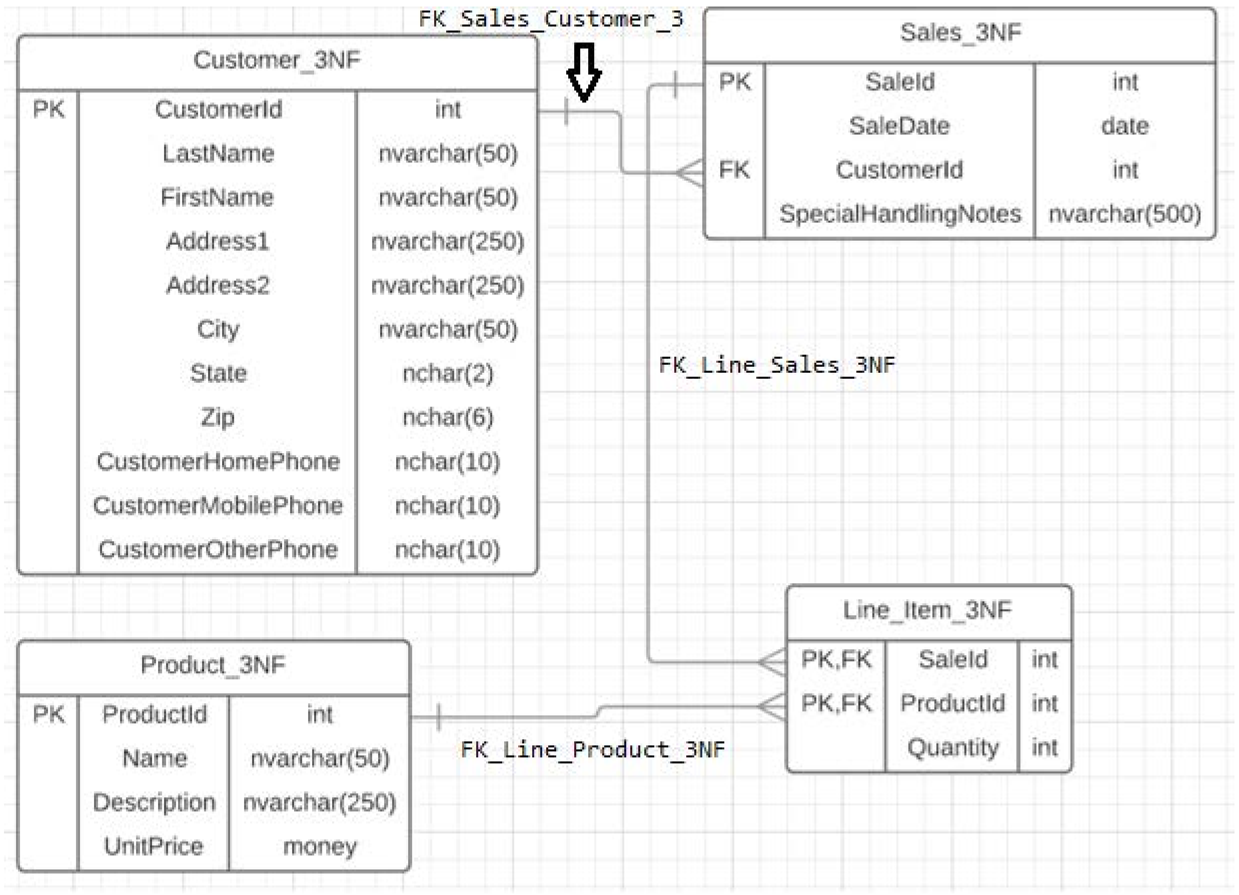
Reasoning

Most of the tables in this form are nearly the same as the second normal form. However, in the third normal form we have added an additional table called Customer. This table breaks out the customer data that pertains to a sale to its own table. Adding this table allows a customer to exist on many orders on the sales table and be linked to a master customer record through the CustomerId column. CustomerId in the Sales table was altered to be a foreign key to enforce valid data. The Line Item table contains a foreign key constraint against the order table to only allow actual order records to be linked to a line item. It also contains a foreign key constraint against the product table to ensure we have only valid products linked to an order. Line Item table also uses a composite primary key utilizing the SaleId and ProductId to enforce unique records. All tables use named primary keys for easier identification than the auto generated key names that SQL uses.

Duncan Nisbett
Student ID: #000856539
C170: Data Management - Applications

Part B Entity Relationship Diagram

Diagram



Customer_3NF		
PK	CustomerId	int
	LastName	nvarchar(50)
	FirstName	nvarchar(50)
	Address1	nvarchar(250)
	Address2	nvarchar(250)
	City	nvarchar(50)
	State	nchar(2)
	Zip	nchar(6)
	CustomerHomePhone	nchar(10)
	CustomerMobilePhone	nchar(10)
	CustomerOtherPhone	nchar(10)

Sales_3NF		
PK	SaleId	int
	SaleDate	date
FK	CustomerId	int
	SpecialHandlingNotes	nvarchar(500)

Line_Item_3NF		
PK,FK	SaleId	int
PK,FK	ProductId	int
	Quantity	int

Product_3NF		
PK	ProductId	int
	Name	nvarchar(50)
	Description	nvarchar(250)
	UnitPrice	money

Explanation

ER Diagram includes 4 tables; Customer, Product, Sales, and Line Item table. These tables combined can be used to derive a complete sales record in its entirety. Removing any one of these tables would create an incomplete set of data for a sale, thus all are required as part of this diagram.

Customer

The customer table is linked to the sales table as per the order form there is only one customer per order. This establishes a one-to-many relationship between customers and sales. A sale must have only one customer, but a customer can have many sales.

Sales

A sale will contain 1 or more items sold. A sale will only have one instance of a customer linked to it along with additional attributes such as the sale date and any special handling notes.

Line Item

The sale items are stored in the line item table. As a sale can have multiple line items, but a line item can only be linked to one sale this establishes a many-to-one relationship between sales and line items. The line item table is a ternary table as it intersects the data between a sale and a product. To enforce unique data the line item table contains a composite primary key making up the SaleId and ProductId columns.

Product

Finally, we have the Product table which is tied to the line items. There can only be one instance of a donut in the product table, but that donut can be ordered many times in the line item table which indicates a one-to-many relationship between products and line items.

Foreign/Primary Keys

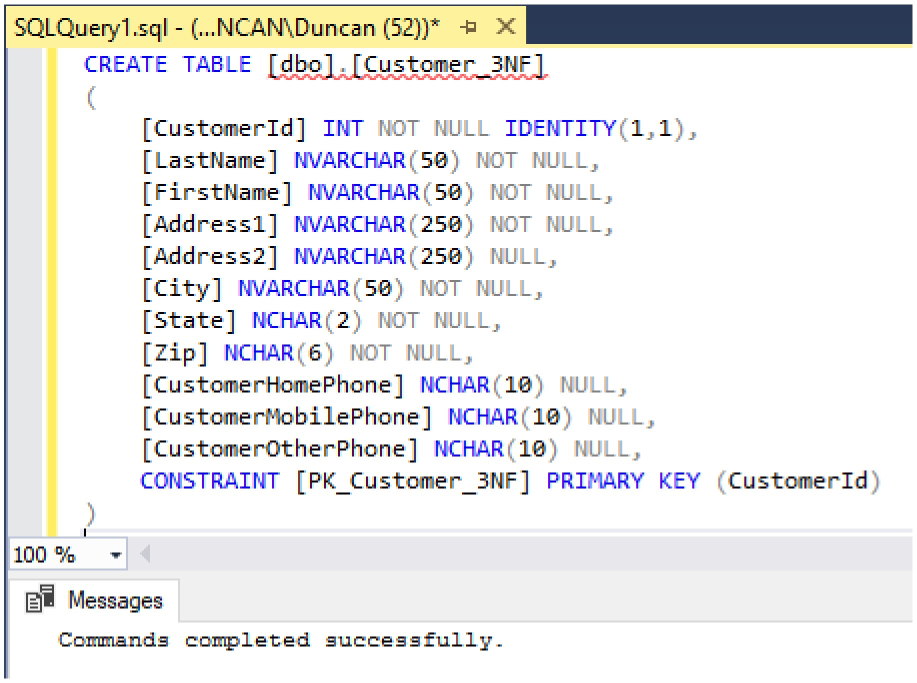
Cardinality is enforced through restrictive foreign key constraints. FK_Sales_Customer_3 requires a valid customer to be required in the Sales table. FK_Line_Sales_3NF requires that any line items are linked to a valid sale record. FK_Line_Product_3NF requires that any product listed in the Line Item table exists within the Product table. Adding a composite key including the SaleId and ProductId enforces a unique constraint which does not allow duplicate products on the same order.

Part C Proof of Compiling

SQL Code

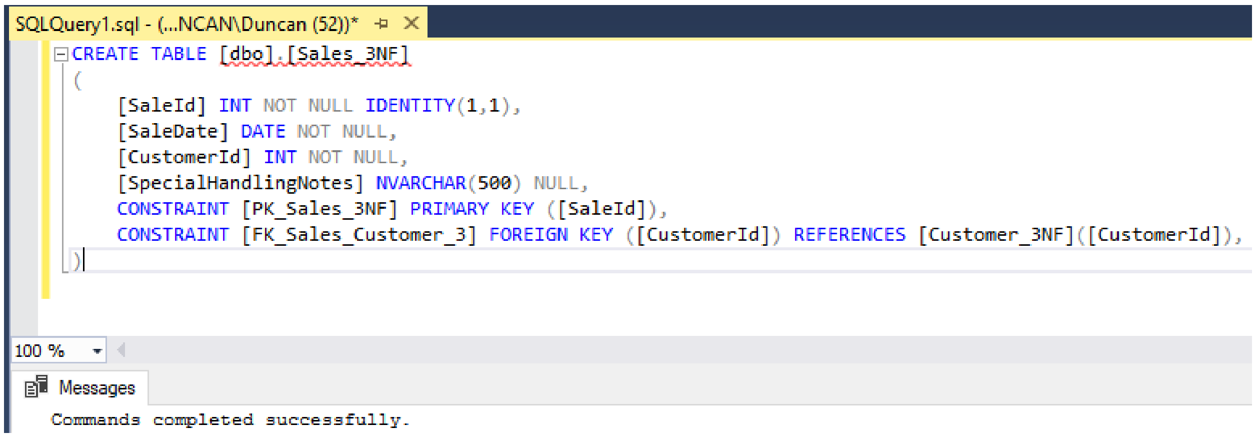
```
CREATE TABLE [dbo].[Customer_3NF]
(
	[CustomerId] INT NOT NULL IDENTITY(1,1),
	[LastName] NVARCHAR(50) NOT NULL,
	[FirstName] NVARCHAR(50) NOT NULL,
	[Address1] NVARCHAR(250) NOT NULL,
	[Address2] NVARCHAR(250) NULL,
	[City] NVARCHAR(50) NOT NULL,
	[State] NCHAR(2) NOT NULL,
	[Zip] NCHAR(6) NOT NULL,
	[CustomerHomePhone] NCHAR(10) NULL,
	[CustomerMobilePhone] NCHAR(10) NULL,
	[CustomerOtherPhone] NCHAR(10) NULL,
	CONSTRAINT [PK_Customer_3NF] PRIMARY KEY (CustomerId)
)


CREATE TABLE [dbo].[Product_3NF]
(
	[ProductId] INT NOT NULL IDENTITY(1,1),
	[Name] NVARCHAR(50) NOT NULL,
	[Description] NVARCHAR(250) NOT NULL,
	[UnitPrice] MONEY NOT NULL,
	CONSTRAINT [PK_Product_3NF] PRIMARY KEY (ProductId)
)


CREATE TABLE [dbo].[Sales_3NF]
(
	[SaleId] INT NOT NULL IDENTITY(1,1),
	[SaleDate] DATE NOT NULL,
	[CustomerId] INT NOT NULL,
	[SpecialHandlingNotes] NVARCHAR(500) NULL,
	CONSTRAINT [PK_Sales_3NF] PRIMARY KEY ([SaleId]),
	CONSTRAINT [FK_Sales_Customer_3] FOREIGN KEY ([CustomerId]) REFERENCES
[Customer_3NF]([CustomerId]),
)


CREATE TABLE [dbo].[Line_Item_3NF]
(
	[SaleId] INT NOT NULL,
	[ProductId] INT NOT NULL,
	[Quantity] INT NOT NULL
	CONSTRAINT [PK_Line_3NF] PRIMARY KEY ([ProductId],[SaleId]),
	CONSTRAINT [FK_Line_Product_3NF] FOREIGN KEY ([ProductId]) REFERENCES
[Product_3NF]([ProductId]),
	CONSTRAINT [FK_Line_Sales_3NF] FOREIGN KEY ([SaleId]) REFERENCES
[Sales_3NF]([SaleId])
)
```

Screenshot Proof



```
CREATE TABLE [dbo].[Customer_3NF]
(
    [CustomerId] INT NOT NULL IDENTITY(1,1),
    [LastName] NVARCHAR(50) NOT NULL,
    [FirstName] NVARCHAR(50) NOT NULL,
    [Address1] NVARCHAR(250) NOT NULL,
    [Address2] NVARCHAR(250) NULL,
    [City] NVARCHAR(50) NOT NULL,
    [State] NCHAR(2) NOT NULL,
    [Zip] NCHAR(6) NOT NULL,
    [CustomerHomePhone] NCHAR(10) NULL,
    [CustomerMobilePhone] NCHAR(10) NULL,
    [CustomerOtherPhone] NCHAR(10) NULL,
    CONSTRAINT [PK_Customer_3NF] PRIMARY KEY (CustomerId)
)
```

```
Commands completed successfully.
```



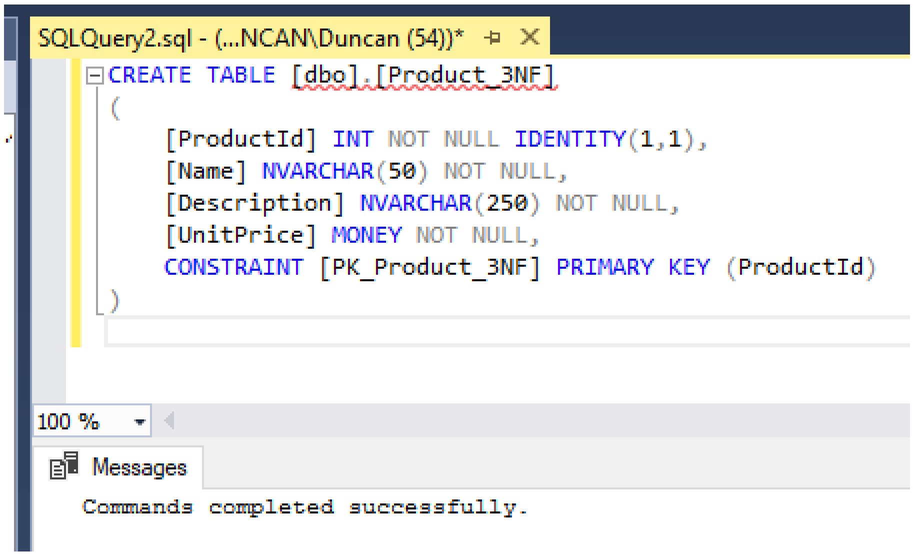
SQLQuery1.sql - (...NCAN\Duncan (52))*

```
CREATE TABLE [dbo].[Sales_3NF]
(
    [SaleId] INT NOT NULL IDENTITY(1,1),
    [SaleDate] DATE NOT NULL,
    [CustomerId] INT NOT NULL,
    [SpecialHandlingNotes] NVARCHAR(500) NULL,
    CONSTRAINT [PK_Sales_3NF] PRIMARY KEY ([SaleId]),
    CONSTRAINT [FK_Sales_Customer_3] FOREIGN KEY ([CustomerId]) REFERENCES [Customer_3NF]([CustomerId]),
)
```

100 %

Messages

```
Commands completed successfully.
```



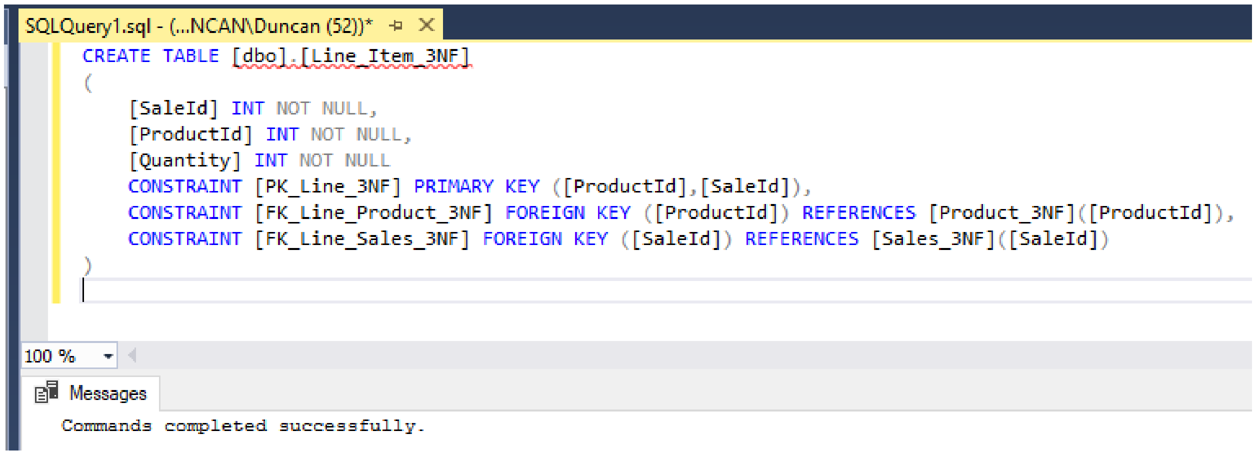
SQLQuery2.sql - (...NCAN\Duncan (54))*

```
CREATE TABLE [dbo].[Product_3NF]
(
    [ProductId] INT NOT NULL IDENTITY(1,1),
    [Name] NVARCHAR(50) NOT NULL,
    [Description] NVARCHAR(250) NOT NULL,
    [UnitPrice] MONEY NOT NULL,
    CONSTRAINT [PK_Product_3NF] PRIMARY KEY (ProductId)
)
```

100 %

Messages

```
Commands completed successfully.
```



SQLQuery1.sql - (...NCAN\Duncan (52))*

```
CREATE TABLE [dbo].[Line_Item_3NF]
(
    [SaleId] INT NOT NULL,
    [ProductId] INT NOT NULL,
    [Quantity] INT NOT NULL
    CONSTRAINT [PK_Line_3NF] PRIMARY KEY ([ProductId],[SaleId]),
    CONSTRAINT [FK_Line_Product_3NF] FOREIGN KEY ([ProductId]) REFERENCES [Product_3NF]([ProductId]),
    CONSTRAINT [FK_Line_Sales_3NF] FOREIGN KEY ([SaleId]) REFERENCES [Sales_3NF]([SaleId])
)
```

100 %

Messages

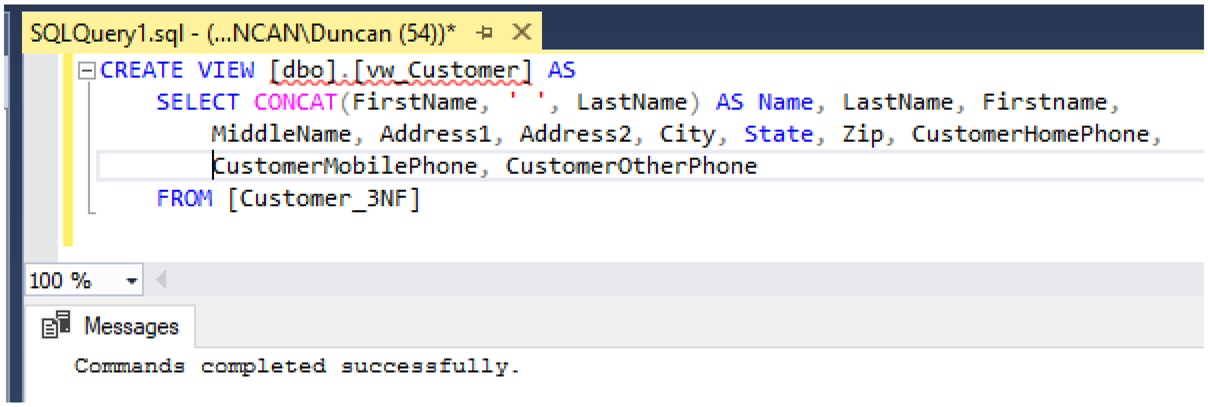
```
Commands completed successfully.
```

Part D Customer View

SQL Code

```
CREATE VIEW [dbo].[vw_Customer] AS
	SELECT CONCAT(FirstName, ' ', LastName) AS Name, LastName, Firstname, Address1,
Address2, City, State, Zip, CustomerHomePhone, CustomerMobilePhone, CustomerOtherPhone
	FROM [Customer_3NF]
```

Screenshot Proof



```
SQLQuery1.sql - (...NCAN\Duncan (54))*
CREATE VIEW [dbo].[vw_Customer] AS
    SELECT CONCAT(FirstName, ' ', LastName) AS Name, LastName, Firstname,
        MiddleName, Address1, Address2, City, State, Zip, CustomerHomePhone,
        CustomerMobilePhone, CustomerOtherPhone
    FROM [Customer_3NF]
```

100 %

Messages

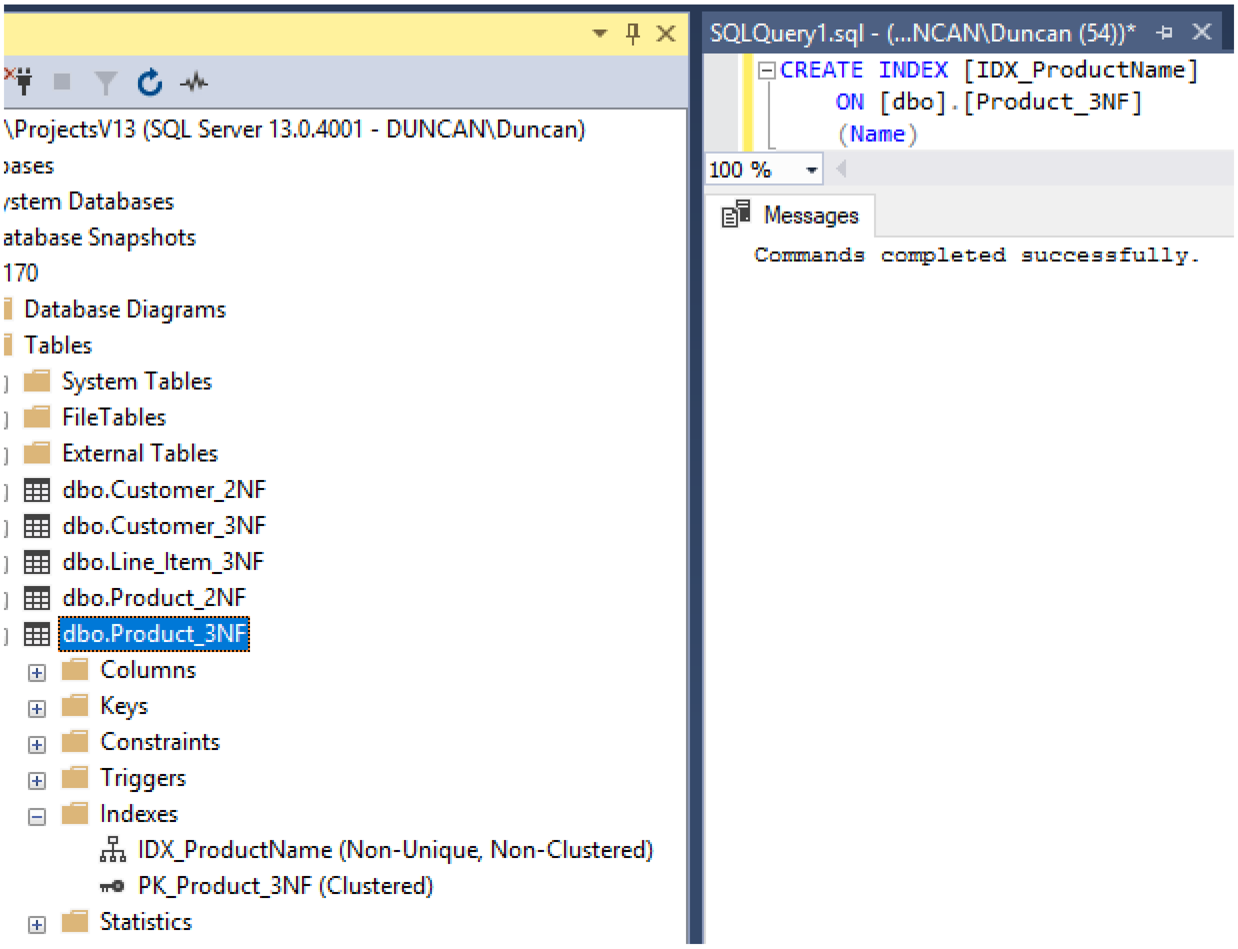
```
Commands completed successfully.
```

Part E Create Product Name Index

SQL Code

```
CREATE INDEX [IDX_ProductName]
	ON [dbo].[Product_3NF]
	(Name)
```

Screenshot Proof


\ProjectsV13 (SQL Server 13.0.4001 - DUNCAN\Duncan)
)ases
/stem Databases
atabase Snapshots
170
Database Diagrams
Tables
System Tables
FileTables
External Tables
dbo.Customer_2NF
dbo.Customer_3NF
dbo.Line_Item_3NF
dbo.Product_2NF
dbo.Product_3NF
Columns
Keys
Constraints
Triggers
Indexes
IDX_ProductName (Non-Unique, Non-Clustered)
PK_Product_3NF (Clustered)
Statistics

SQLQuery1.sql - (...NCAN\Duncan (54))*

```
CREATE INDEX [IDX_ProductName]
    ON [dbo].[Product_3NF]
    (Name)
```

100 %

Messages

```
Commands completed successfully.
```

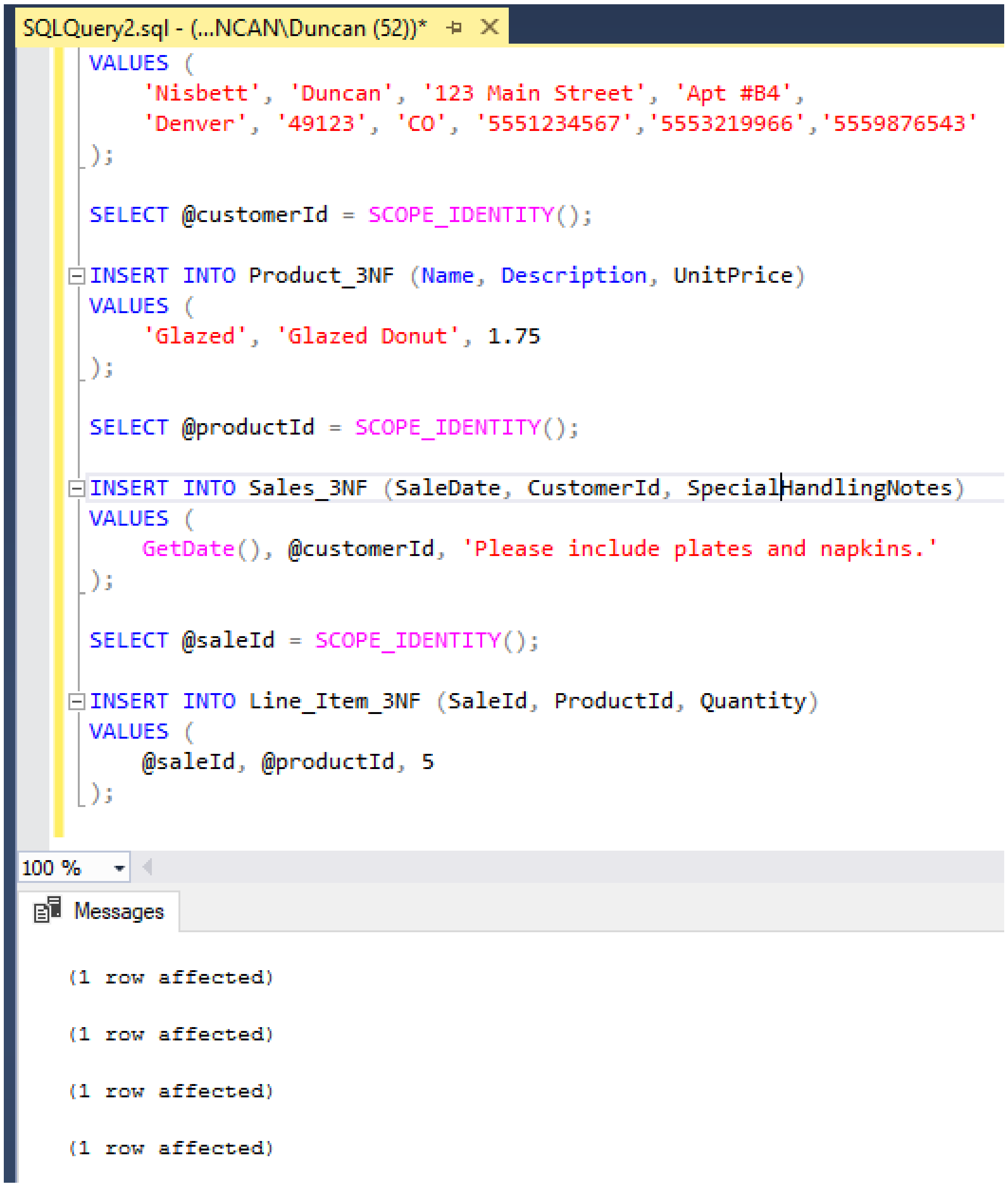
Part F Populate Tables

SQL Code

```
DECLARE @customerId int;
DECLARE @productId int;
DECLARE @saleId int;

INSERT INTO Customer_3NF (LastName, FirstName, Address1, Address2,
       City, Zip, State, CustomerHomePhone, CustomerMobilePhone, CustomerOtherPhone)
VALUES (
       'Nisbett', 'Duncan', '123 Main Street', 'Apt #B4',
       'Denver', '49123', 'CO', '5551234567','5553219966','5559876543'
);

SELECT @customerId = SCOPE_IDENTITY();

INSERT INTO Product_3NF (Name, Description, UnitPrice)
VALUES (
       'Glazed', 'Glazed Donut', 1.75
);
```

```
SELECT @productId = SCOPE_IDENTITY();

INSERT INTO Sales_3NF (SaleDate, CustomerId, SpecialHandlingNotes)
VALUES (
	GetDate(), @customerId, 'Please include plates and napkins.'
);

SELECT @saleId = SCOPE_IDENTITY();

INSERT INTO Line_Item_3NF (SaleId, ProductId, Quantity)
VALUES (
	@saleId, @productId, 5
);
```

Screenshot Proof



SQLQuery2.sql - (...NCAN\Duncan (52))*

```
VALUES (
    'Nisbett', 'Duncan', '123 Main Street', 'Apt #B4',
    'Denver', '49123', 'CO', '5551234567','5553219966','5559876543'
);

SELECT @customerId = SCOPE_IDENTITY();

INSERT INTO Product_3NF (Name, Description, UnitPrice)
VALUES (
    'Glazed', 'Glazed Donut', 1.75
);

SELECT @productId = SCOPE_IDENTITY();

INSERT INTO Sales_3NF (SaleDate, CustomerId, SpecialHandlingNotes)
VALUES (
    GetDate(), @customerId, 'Please include plates and napkins.'
);

SELECT @saleId = SCOPE_IDENTITY();

INSERT INTO Line_Item_3NF (SaleId, ProductId, Quantity)
VALUES (
    @saleId, @productId, 5
);
```

100 %

Messages

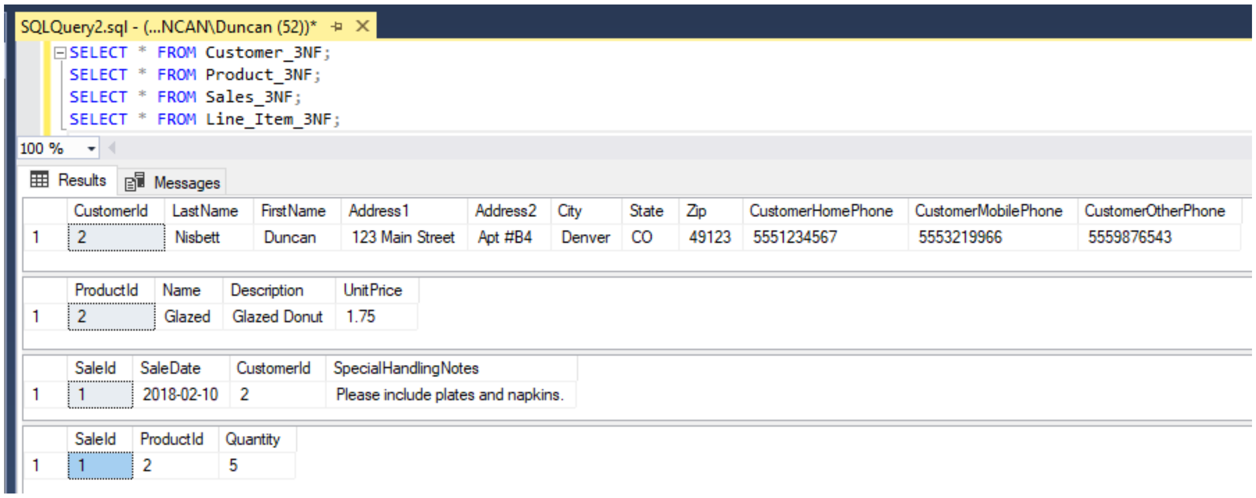
```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

Part G Display Values and Complex Join

Queries

```sql
SELECT * FROM Customer_3NF;
SELECT * FROM Product_3NF;
SELECT * FROM Sales_3NF;
SELECT * FROM Line_Item_3NF;
```

Screenshot Proof



SQLQuery2.sql - (...NCAN\Duncan (52))*

```sql
SELECT * FROM Customer_3NF;
SELECT * FROM Product_3NF;
SELECT * FROM Sales_3NF;
SELECT * FROM Line_Item_3NF;
```

100 %

Results | Messages

	CustomerId	LastName	FirstName	Address1	Address2	City	State	Zip	CustomerHomePhone	CustomerMobilePhone	CustomerOtherPhone
1	2	Nisbett	Duncan	123 Main Street	Apt #B4	Denver	CO	49123	5551234567	5553219966	5559876543

	ProductId	Name	Description	UnitPrice
1	2	Glazed	Glazed Donut	1.75

	SaleId	SaleDate	CustomerId	SpecialHandlingNotes
1	1	2018-02-10	2	Please include plates and napkins.

	SaleId	ProductId	Quantity
1	1	2	5

Complex Join SQL

Quick

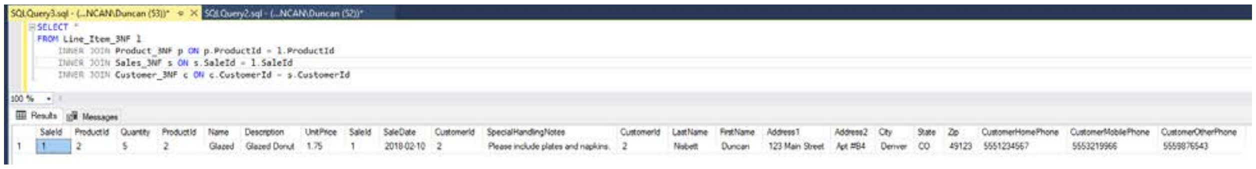
```sql
SELECT *
FROM Line_Item_3NF l
	INNER JOIN Product_3NF p ON p.ProductId = l.ProductId
	INNER JOIN Sales_3NF s ON s.id = l.OrderId
	INNER JOIN Customer_3NF c ON c.CustomerId = s.CustomerId
```

Proper

```sql
SELECT s.SaleDate, s.SaleId, c.CustomerId, c.FirstName, c.LastName, c.Address1,
	c.Address2, c.City, c.State, c.Zip, c.CustomerHomePhone, c.CustomerMobilePhone,
c.CustomerOtherPhone,
	l.Quantity, p.ProductId, p.Name, p.Description, p.UnitPrice,
s.SpecialHandlingNotes
FROM Line_Item_3NF l
	INNER JOIN Sales_3NF s ON s.SaleId = l.SaleId
	INNER JOIN Product_3NF p ON p.ProductId = l.ProductId
	INNER JOIN Customer_3NF c ON c.CustomerId = s.CustomerId
```

Complex Join Screenshot

Quick



SQLQuery3.sql - (...NCAN\Duncan (53))* | SQLQuery2.sql - (...NCAN\Duncan (52))*

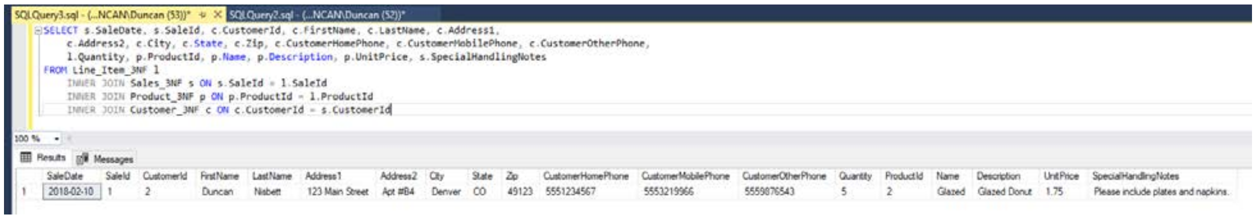
```sql
SELECT *
FROM Line_Item_3NF l
	INNER JOIN Product_3NF p ON p.ProductId = l.ProductId
	INNER JOIN Sales_3NF s ON s.SaleId = l.SaleId
	INNER JOIN Customer_3NF c ON c.CustomerId = s.CustomerId
```

100 %

Results | Messages

	SaleId	ProductId	Quantity	ProductId	Name	Description	UnitPrice	SaleId	SaleDate	CustomerId	SpecialHandlingNotes	CustomerId	LastName	FirstName	Address1	Address2	City	State	Zip	CustomerHomePhone	CustomerMobilePhone	CustomerOtherPhone
1	1	2	5	2	Glazed	Glazed Donut	1.75	1	2018-02-10	2	Please include plates and napkins.	2	Nisbett	Duncan	123 Main Street	Apt #B4	Denver	CO	49123	5551234567	5553219966	5559876543

Proper


SQLQuery3.sql - (...NCAN\Duncan (53))*
SQLQuery2.sql - (...NCAN\Duncan (52))*
100 %
Results
Messages

```sql
SELECT s.SaleDate, s.SaleId, c.CustomerId, c.FirstName, c.LastName, c.Address1,
    c.Address2, c.City, c.State, c.Zip, c.CustomerHomePhone, c.CustomerMobilePhone, c.CustomerOtherPhone,
    l.Quantity, p.ProductId, p.Name, p.Description, p.UnitPrice, s.SpecialHandlingNotes
FROM Line_Item_3NF l
    INNER JOIN Sales_3NF s ON s.SaleId = l.SaleId
    INNER JOIN Product_3NF p ON p.ProductId = l.ProductId
    INNER JOIN Customer_3NF c ON c.CustomerId = s.CustomerId
```

	SaleDate	SaleId	CustomerId	FirstName	LastName	Address1	Address2	City	State	Zip	CustomerHomePhone	CustomerMobilePhone	CustomerOtherPhone	Quantity	ProductId	Name	Description	UnitPrice	SpecialHandlingNotes
1	2018-02-10	1	2	Duncan	Nisbett	123 Main Street	Apt #B4	Denver	CO	49123	5551234567	5553219966	5559876543	5	2	Glazed	Glazed Donut	1.75	Please include plates and napkins.