Kyle Santana
Student ID: 311788
Data Management - Applications – C170
Performance Assessment

Contents

A.

First Normal Form 1A

To transform our unnormalized data into the first normal form, we first must create a relation that uses a composite key, then enter new Order data for each customer into a new row. For our Orders example the Order ID and Donut ID attributes will become the composite primary key. The table below shows this approach applied to the Orders relation and saved as one table per NF1 requirements.

Orders Table:

Order ID (PK), Customer ID, Customer Last Name, Customer First Name, Street Address, Apt No, City, State, Zip, Home Phone, Mobile Phone, Other Phone, Notes, Order Date, Donut ID (PK), Donut Description, Donut Name, Unit Price, Quantity

Second Normal Form 1B

In order to convert a relation into 2NF it must first be in 1NF which it currently is and all of the non-key attributes are functionally dependent on the entire primary key.

Looking again at the 1NF relation in our Orders example:

In the Orders relation, we can see that Customer ID, Customer Last Name, Customer First Name, Street Address, Apt No, City, State, Zip, Home Phone, Mobile Phone, Other Phone, Notes, Order Date and Order Total depends only on the Order ID instead of the combination of Order ID and Donut ID. These fields are said to be functionally dependent on Order ID because at any point in time, there can be only one value associated with a given Order ID; however, because there is no dependency on Donut ID, which is a part of the primary key, these fields are in violation of 2NF.

Looking at the Donut Description or Donut Name fields we determine that Donut ID alone determines the value. Basically the same Donut can appear as a line item on many different orders, the Donut Description or Donut Name is the same regardless of the Order ID. We can say that Donut Description or Donut Name is functionally dependent on only part of the primary key because it depends only on Donut ID and

not on the combination of Order ID and Donut ID. This partial dependence is the very issue 2NF addresses. The Quantity field in the 1NF Orders relation is the only one that depends on the combination of Order ID and Donut ID

Orders Table:
Order ID(PK), Customer ID, Customer Last Name, Customer First Name, Street Address, Apt No, City, State, Zip, Home Phone, Mobile Phone, Other Phone, Notes, Order Date

Line Item Table:
Order ID(PK)(FK), Donut ID(PK)(FK), Quantity

Donut Table:
Donut ID (PK), Donut Description, Donut Name, Unit Price

Third Normal Form 1C Screenshot

To convert from 2NF to 3NF, we must first find any transitive dependencies which is an attribute that depends on another attribute that is not the primary key of the relation. Looking at our Orders relation in 2NF, we see Customer First Name is dependent on the Order ID (each Order ID has only one Customer Last Name value associated with it), but at the same time, Customer Last Name is also dependent on Customer Number. The same can be said of the rest of the customer attributes as well. The problem here is that attributes of the Customer entity have been included in our Orders relation. In order for us to complete the transformation of our 2NF relation into 3NF we must move the transitively dependent attributes to a relation where they are dependent only on the primary key. As I move the attributes I will also leave the dependent attributes in the original relation as a foreign key.

Donut Table:
Donut ID (PK), Donut Description, Donut Name, Unit Price

Orders Table:
Order ID (PK), Notes, Order Date, Customer ID(FK)

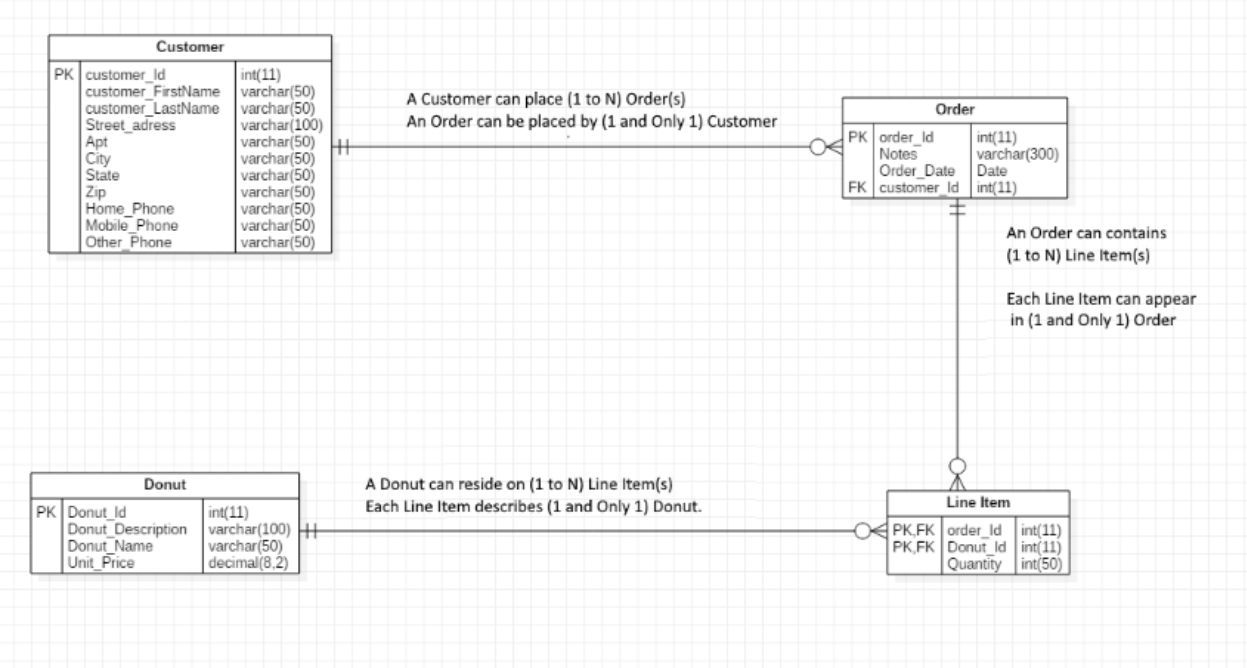
Line Item Table:
Order ID(PK)(FK), Donut ID(PK)(FK), Quantity

Customer Table:
Customer ID (PK), Customer Last Name, Customer First Name, Street Address, Apt No, City, State, Zip, Home Phone, Mobile Phone, Other Phone.

B.

1-3. Below is an Entity Relationship Diagram that represents each of the tables from the previous question in third normalized form with the keys and data types illustrated.

B1-3 Screenshot



4. Entity Selection Decision Criteria

AB. Each of the entities in the ER diagram were chosen because they each represent one of the logical tables that will be present in the database. All the logical tables combined form a complete sales history for Donuts-R-Us. The Cardinalities (relationships) were chosen based off the following rules/actions derived from the invoice example that was provided:

A Customer can place (1 to N) Order(s)

An Order can be placed by (1 and Only 1) Customer

An Order can contains (1 to N) Line Item(s)

Each Line Item can appear in (1 and Only 1) Order

A Donut can reside on (1 to N) Line Item(s)

Each Line Item describes (1 and Only 1) Donut.

Using the previous determined rules we know that the Customer and Orders table has a one-to-many (1:M) relationship since one customer can place many orders. We know that Orders and Line Item has a one-to-many (1:M) relationship since a customer has the ability to purchase more than one donut in one order. We also know that Donut and Orders have a one-to-many (1:M) relationship since one type of donut can be ordered from many customers.

C. Cardinality was determined by looking at the sales order sheet and determining that 1 customer can place multiple orders. 1 order can contain multiple donuts but only 1 customer. To return an order with multiple different types of donuts we need the item line table which can have multiple lines in an order.

C.

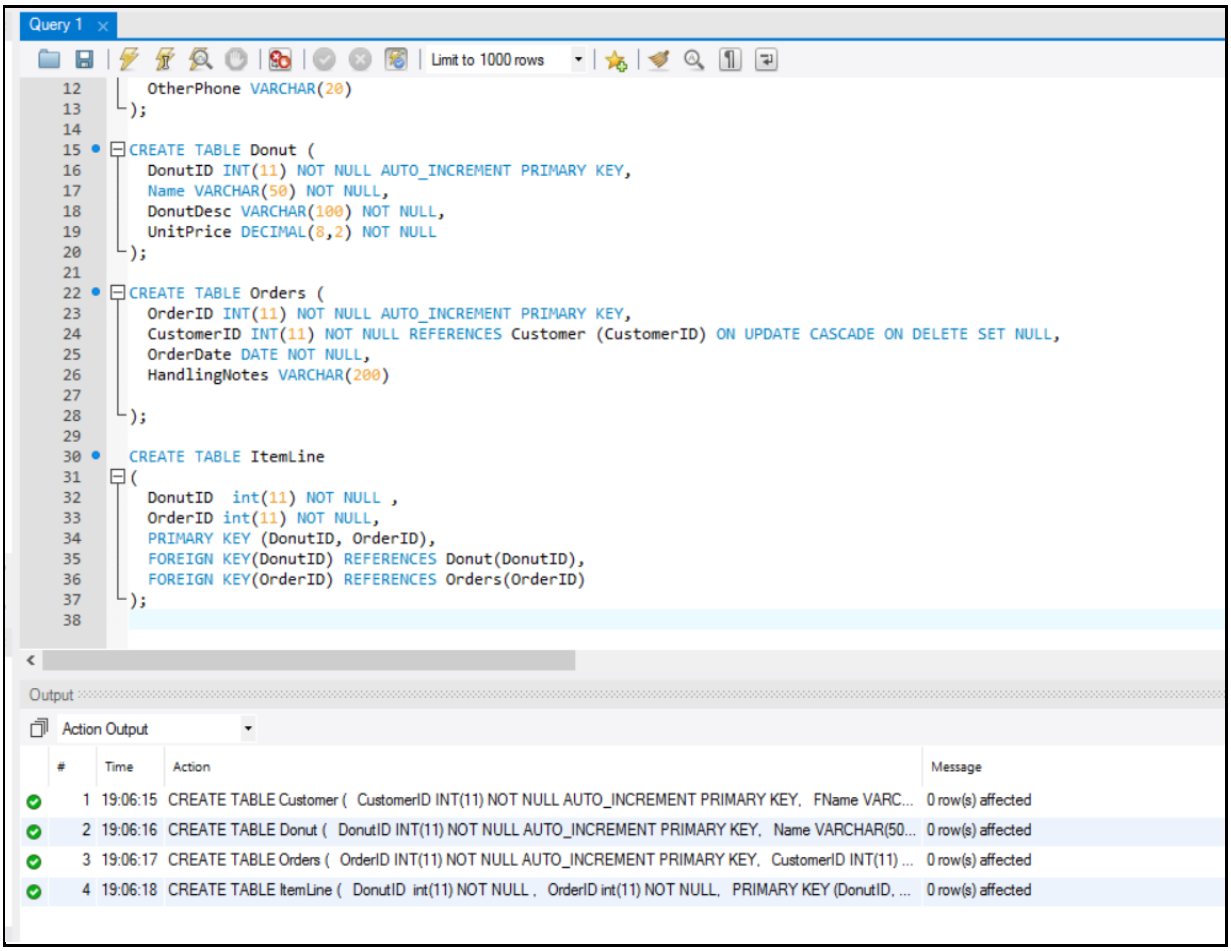
1. SQL code to create each of the third normal form tables from previous question.

C1 Code:

```
CREATE TABLE Customer (
  CustomerID INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  FName VARCHAR(50) NOT NULL,
  LName VARCHAR(50) NOT NULL,
  Address VARCHAR(100) NOT NULL,
  Apt VARCHAR(50),
  City VARCHAR(50) NOT NULL,
  State VARCHAR(50) NOT NULL,
  Zip VARCHAR(50) NOT NULL,
  HomePhone VARCHAR(20),
  PhoneNumber VARCHAR(20),
  OtherPhone VARCHAR(20)
);

CREATE TABLE Donut (
  DonutID INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(50) NOT NULL,
```

```
  DonutDesc VARCHAR(100) NOT NULL,
  UnitPrice DECIMAL(8,2) NOT NULL
);


CREATE TABLE Orders (
  OrderID INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  CustomerID INT(11) NOT NULL REFERENCES Customer (CustomerID) ON UPDATE
CASCADE ON DELETE SET NULL,
  OrderDate DATE NOT NULL,
  HandlingNotes VARCHAR(200)


);


CREATE TABLE ItemLine
(
  DonutID  int(11) NOT NULL ,
  OrderID int(11) NOT NULL,
  Qty int(11) NOT NULL,
  PRIMARY KEY (DonutID, OrderID),
  FOREIGN KEY(DonutID) REFERENCES Donut(DonutID),
  FOREIGN KEY(OrderID) REFERENCES Orders(OrderID)
);
```

C1A Screenshot:



D.

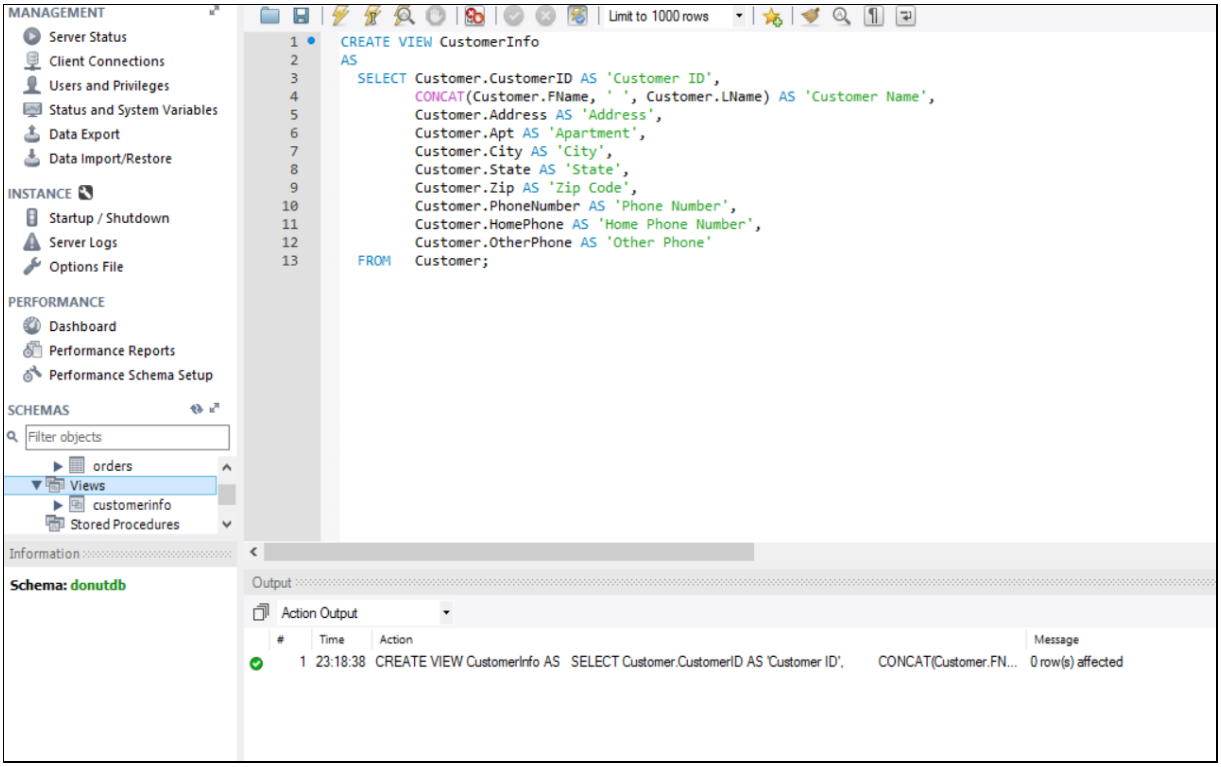
1. SQL code that creates a “View” that shows all of the customer information with the first name and last name concatenated.

D1 Code:

```
CREATE VIEW CustomerInfo
AS
  SELECT Customer.CustomerID AS 'Customer ID',
         CONCAT(Customer.FName, ' ', Customer.LName) AS 'Customer Name',
         Customer.Address AS 'Address',
         Customer.Apt AS 'Apartment',
         Customer.City AS 'City',
         Customer.State AS 'State',
         Customer.Zip AS 'Zip Code',
         Customer.PhoneNumber AS 'Phone Number',
```

```
        Customer.HomePhone AS 'Home Phone Number',
        Customer.OtherPhone AS 'Other Phone'
FROM    Customer;
```

D1A Screenshot:



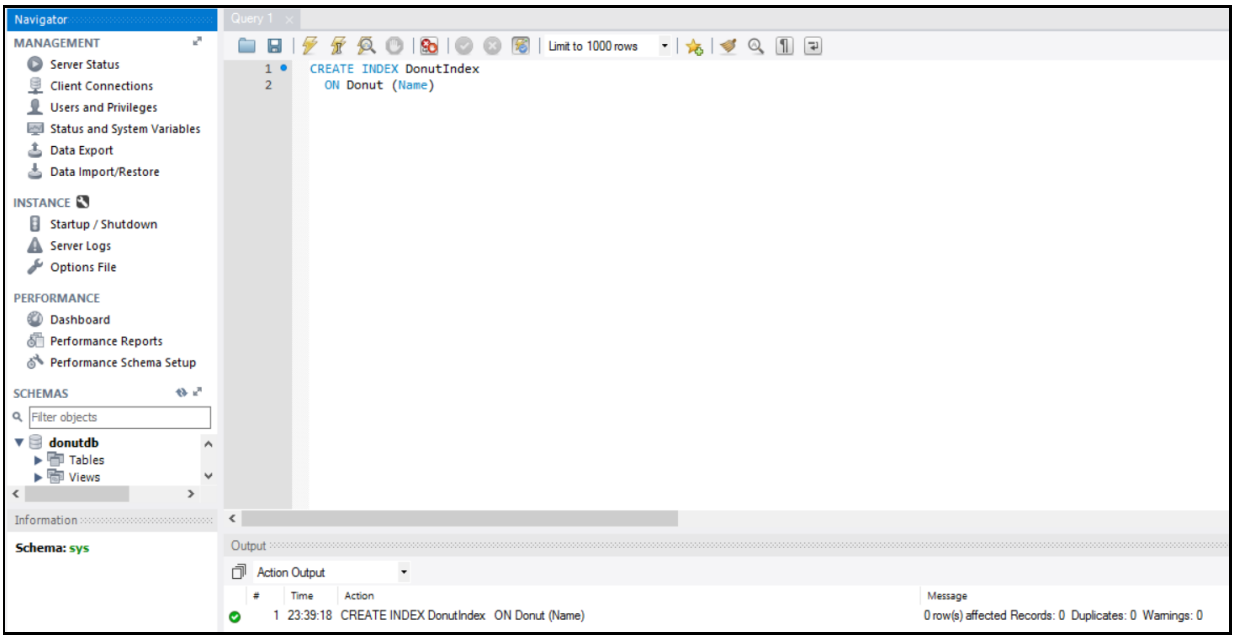
E.

1. SQL code to create an “Index” for the donut information.

E1 Code:

```
CREATE INDEX DonutIndex
  ON Donut (Name)
```

E1A Screenshot:



F.

1. SQL code to populate all of the tables

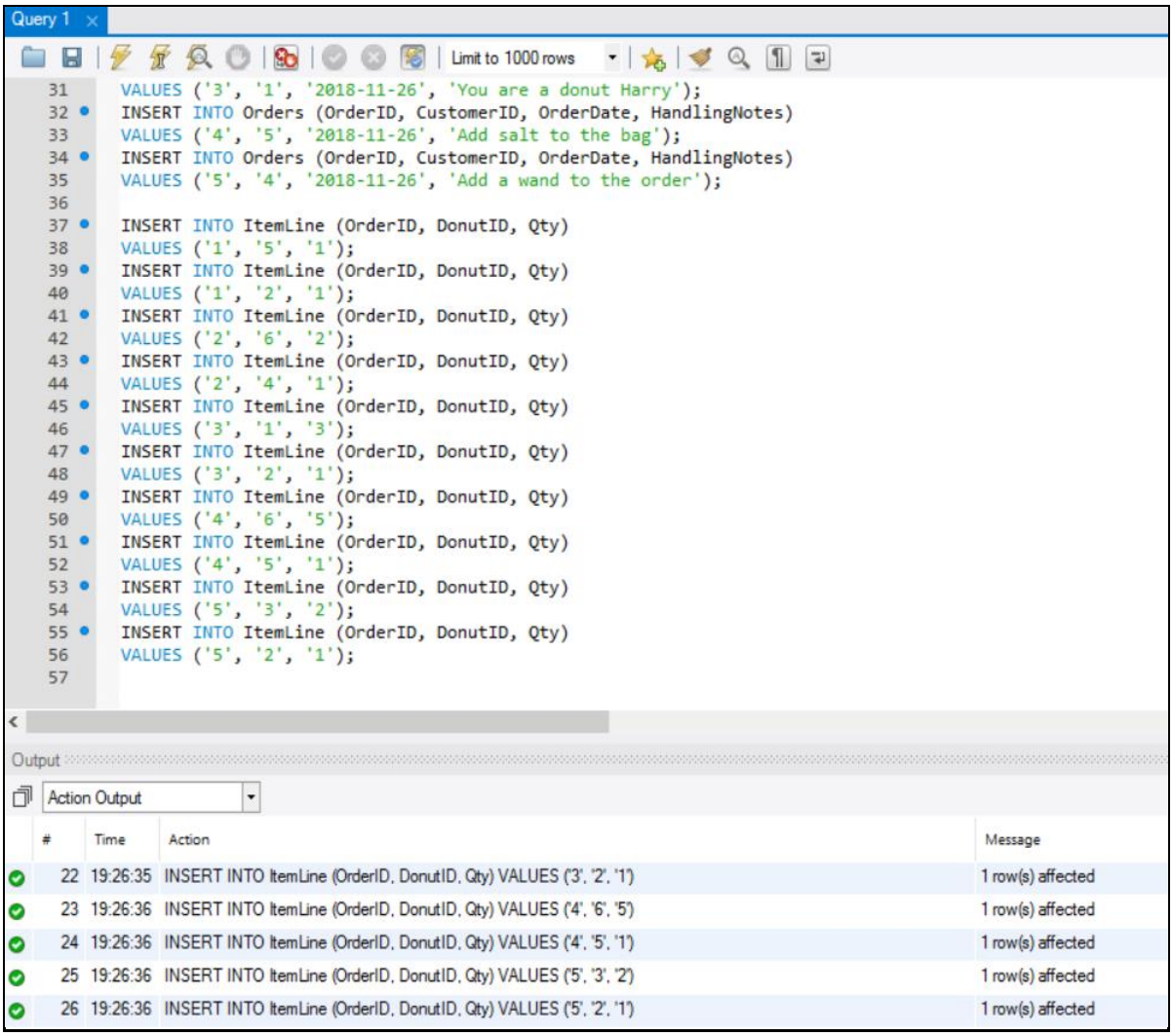
F1 Code:

```
INSERT INTO Customer (CustomerID, FName, LName, Address, Apt, City, State,
Zip, HomePhone, PhoneNumber, OtherPhone)

VALUES ('1', 'Tom', 'Riddle', '123 Evil St.', '100B', 'Alexandria', 'VA',
'22304', '540-555-5555', '540-555-5556', '540-555-5557');

INSERT INTO Customer (CustomerID, FName, LName, Address, Apt, City, State,
Zip, HomePhone, PhoneNumber, OtherPhone)

VALUES ('2', 'Harry', 'Potter', '123 Hero St.', '100C', 'Alexandria', 'VA',
'22304', '540-555-5558', '540-555-5559', '540-555-5560');

INSERT INTO Customer (CustomerID, FName, LName, Address, Apt, City, State,
Zip, HomePhone, PhoneNumber, OtherPhone)

VALUES ('3', 'Hermione', 'Granger', '123 Goblet St.', '200C', 'Alexandria',
'VA', '22304', '540-555-5561', '540-555-5562', '540-555-5563');

INSERT INTO Customer (CustomerID, FName, LName, Address, Apt, City, State,
Zip, HomePhone, PhoneNumber, OtherPhone)

VALUES ('4', 'Lord', 'Voldemort', '123 No Nose St.', '300C', 'Alexandria',
'VA', '22304', '540-555-5564', '540-555-5565', '540-555-5566');
```

```
INSERT INTO Customer (CustomerID, FName, LName, Address, Apt, City, State,
Zip, HomePhone, PhoneNumber, OtherPhone)

VALUES ('5', 'Ron', 'Weasley', '123 Best Bud St.', '130C', 'Alexandria',
'VA', '22304', '540-555-5567', '540-555-5568', '540-555-5569');


INSERT INTO Donut (DonutID, Name, DonutDesc, UnitPrice)

VALUES ('1', 'Plain', 'Plain Donut', 1.50);

INSERT INTO Donut (DonutID, Name, DonutDesc, UnitPrice)

VALUES ('2', 'Glazed', 'Glazed Donut', 1.75);

INSERT INTO Donut (DonutID, Name, DonutDesc, UnitPrice)

VALUES ('3', 'Cinnamon', 'Cinnamon Donut', 1.75);

INSERT INTO Donut (DonutID, Name, DonutDesc, UnitPrice)

VALUES ('4', 'Chocolate', 'Chocolate Donut', 1.75);

INSERT INTO Donut (DonutID, Name, DonutDesc, UnitPrice)

VALUES ('5', 'Sprinkle', 'Sprinkle Donut', 1.75);

INSERT INTO Donut (DonutID, Name, DonutDesc, UnitPrice)

VALUES ('6', 'Gluten-Free', 'Gluten-Free Donut', 2.00);



INSERT INTO Orders (OrderID, CustomerID, OrderDate, HandlingNotes)

VALUES ('1', '2', '2018-11-26', 'Do not forget plates this time');

INSERT INTO Orders (OrderID, CustomerID, OrderDate, HandlingNotes)

VALUES ('2', '3', '2018-11-26', 'My donuts better be fresh or you will
suffer');

INSERT INTO Orders (OrderID, CustomerID, OrderDate, HandlingNotes)

VALUES ('3', '1', '2018-11-26', 'You are a donut Harry');

INSERT INTO Orders (OrderID, CustomerID, OrderDate, HandlingNotes)

VALUES ('4', '5', '2018-11-26', 'Add salt to the bag');

INSERT INTO Orders (OrderID, CustomerID, OrderDate, HandlingNotes)

VALUES ('5', '4', '2018-11-26', 'Add a wand to the order');


INSERT INTO ItemLine (OrderID, DonutID, Qty)

VALUES ('1', '5', '1');
```

```
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('1', '2', '1');
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('2', '6', '2');
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('2', '4', '1');
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('3', '1', '3');
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('3', '2', '1');
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('4', '6', '5');
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('4', '5', '1');
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('5', '3', '2');
INSERT INTO ItemLine (OrderID, DonutID, Qty)
VALUES ('5', '2', '1');
```

F1A Screenshot:



G

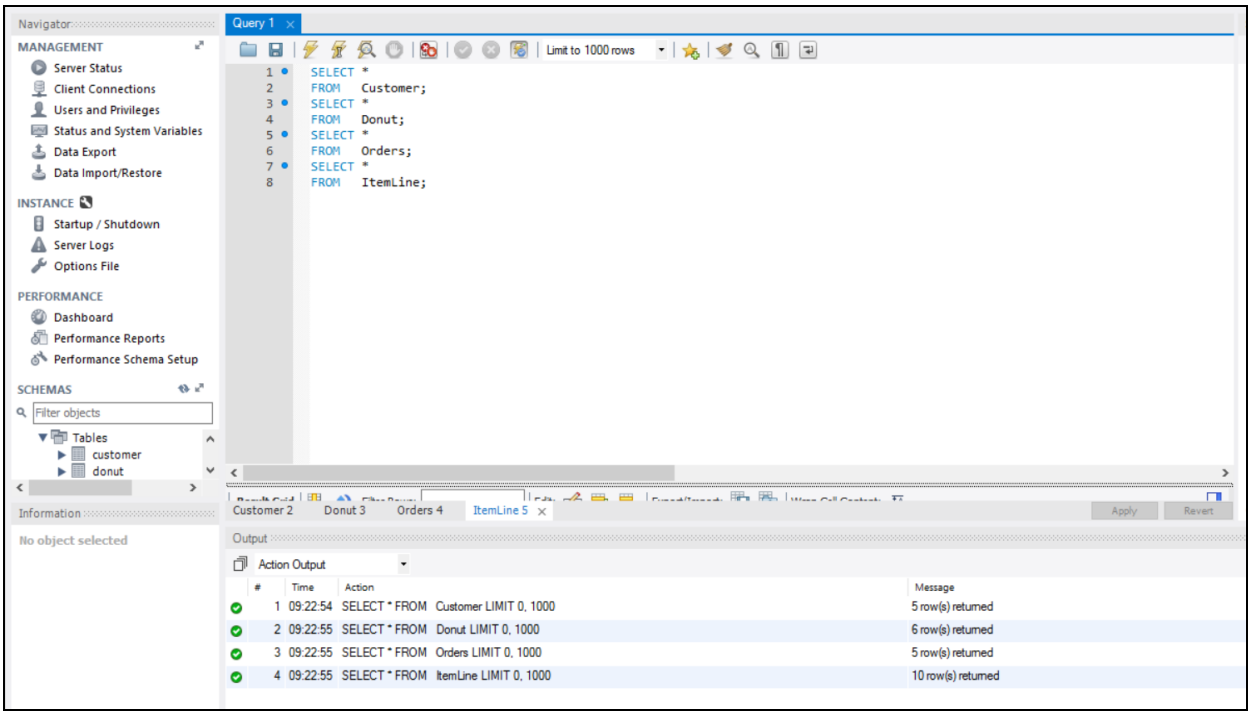
1. SQL code for the simple (sfw) queries to display all of the data in each of the tables that have been created and populated.

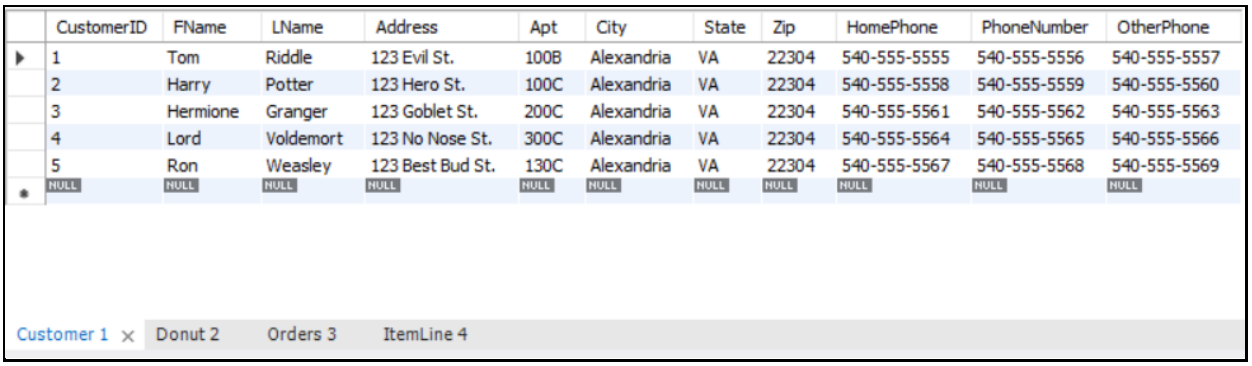
G1 Code:

```
SELECT *
FROM   Customer;

SELECT *
FROM   Donut;

SELECT *
FROM   Orders;
```

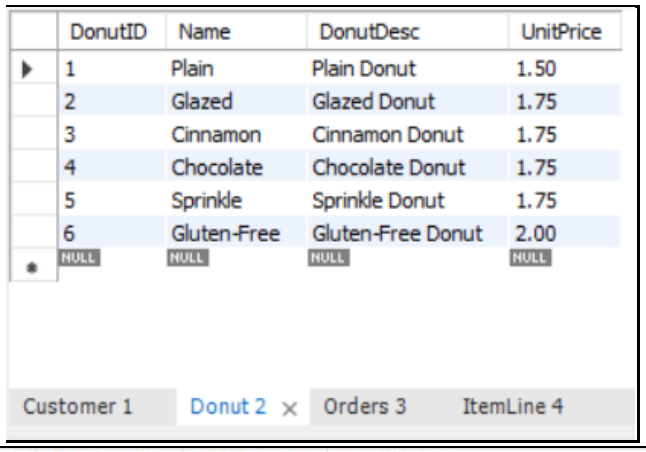
```
SELECT *
FROM   ItemLine;
```

G1A Screenshot:



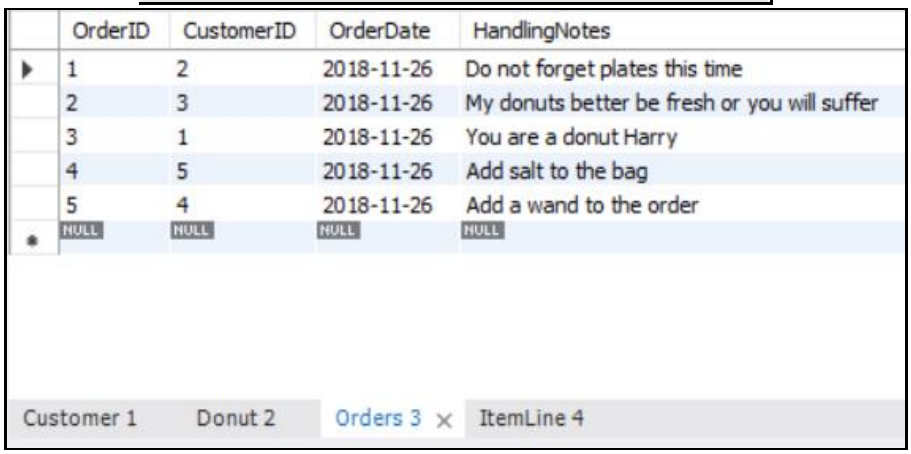


CustomerID	FName	LName	Address	Apt	City	State	Zip	HomePhone	PhoneNumber	OtherPhone
1	Tom	Riddle	123 Evil St.	100B	Alexandria	VA	22304	540-555-5555	540-555-5556	540-555-5557
2	Harry	Potter	123 Hero St.	100C	Alexandria	VA	22304	540-555-5558	540-555-5559	540-555-5560
3	Hermione	Granger	123 Goblet St.	200C	Alexandria	VA	22304	540-555-5561	540-555-5562	540-555-5563
4	Lord	Voldemort	123 No Nose St.	300C	Alexandria	VA	22304	540-555-5564	540-555-5565	540-555-5566
5	Ron	Weasley	123 Best Bud St.	130C	Alexandria	VA	22304	540-555-5567	540-555-5568	540-555-5569
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



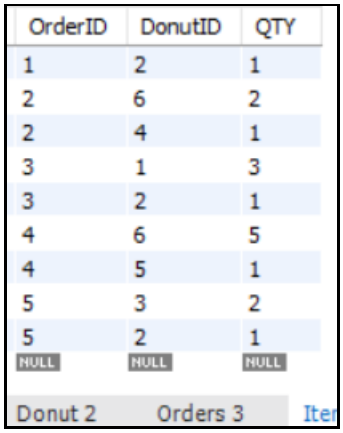
DonutID	Name	DonutDesc	UnitPrice
1	Plain	Plain Donut	1.50
2	Glazed	Glazed Donut	1.75
3	Cinnamon	Cinnamon Donut	1.75
4	Chocolate	Chocolate Donut	1.75
5	Sprinkle	Sprinkle Donut	1.75
6	Gluten-Free	Gluten-Free Donut	2.00
NULL	NULL	NULL	NULL

Customer 1 | Donut 2 | Orders 3 | ItemLine 4



OrderID	CustomerID	OrderDate	HandlingNotes
1	2	2018-11-26	Do not forget plates this time
2	3	2018-11-26	My donuts better be fresh or you will suffer
3	1	2018-11-26	You are a donut Harry
4	5	2018-11-26	Add salt to the bag
5	4	2018-11-26	Add a wand to the order
NULL	NULL	NULL	NULL

Customer 1 | Donut 2 | Orders 3 | ItemLine 4



OrderID	DonutID	QTY
1	2	1
2	6	2
2	4	1
3	1	3
3	2	1
4	6	5
4	5	1
5	3	2
5	2	1
NULL	NULL	NULL

Donut 2 | Orders 3 | Iter

2. SQL code for a complex join query to display all of the information contained in the attached "Sales Order Form"

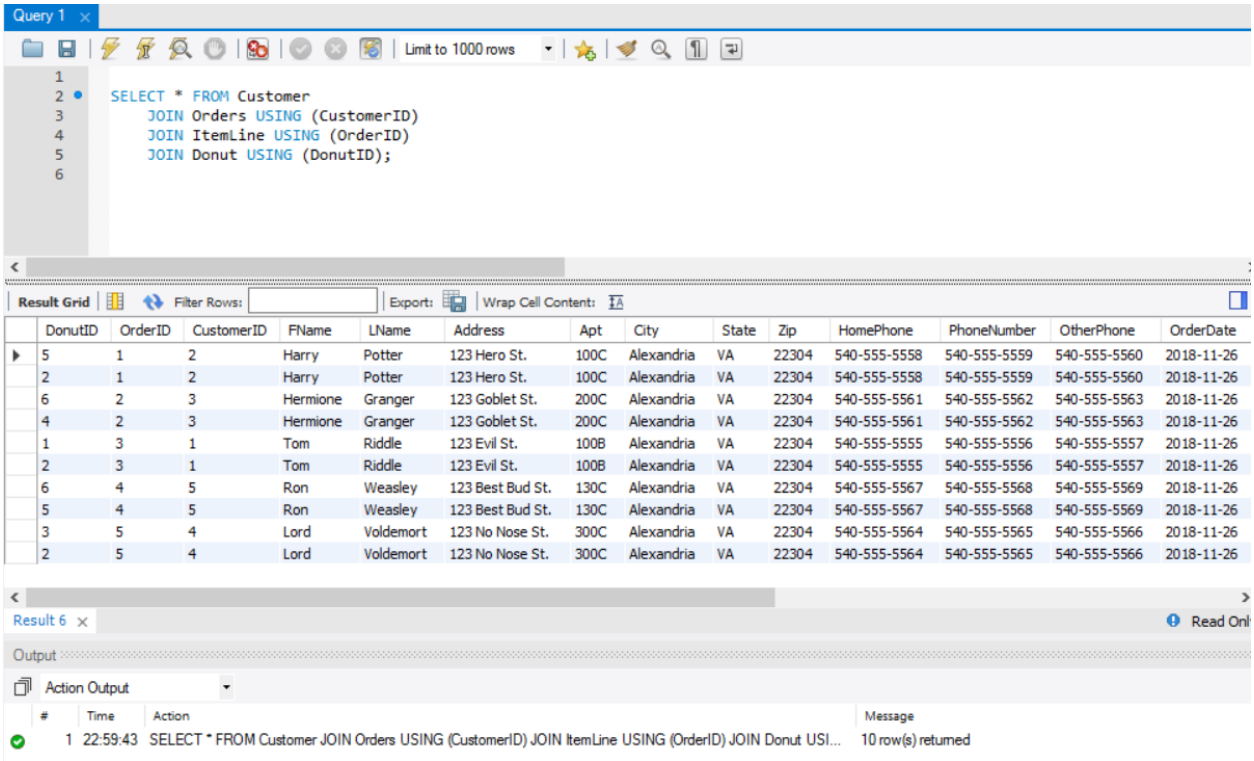
G2 Code:

```
SELECT Customer.*,
       Donut.*,
       Orders.OrderID,
       Orders.OrderDate,
```

```
        ItemLine.Qty
```

```
1.  SELECT * FROM Customer
2.      JOIN Orders USING (CustomerID)
3.      JOIN ItemLine USING (OrderID)
4.      JOIN Donut USING (DonutID);
```

G2A Screenshot:



Sales Order Example

Donuts-R-Us

You eat them fresh, we bake them fresh.

Date: November 27, 2018

Donut Order ID: [1]

Customer ID: [1]

Customer: **[First Name] [Last Name]**

[Street Address] [Apt. #]

[City, ST ZIP Code]

[Home Phone] [Mobile Phone] [Other Phone]

Qty	Donut ID	Name	Description	Unit Price	Line Total
1	1	Plain	Plain Donut	$1.50	$1.50
5	2	Glazed	Glazed Donut	$1.75	$8.75
12	3	Cinnamon	Cinnamon Donut	$1.75	$21.00
3	4	Chocolate	Chocolate Donut	$1.75	$5.25
4	5	Sprinkle	Sprinkle Donut	$1.75	$7.00
5	6	Gluten-Free	Gluten-Free Donut	$2.00	$10.00
				Subtotal	$53.50
				Sales Tax	10%
				Total	**$58.85**

Special Handling Notes:

Please include plates and napkins.