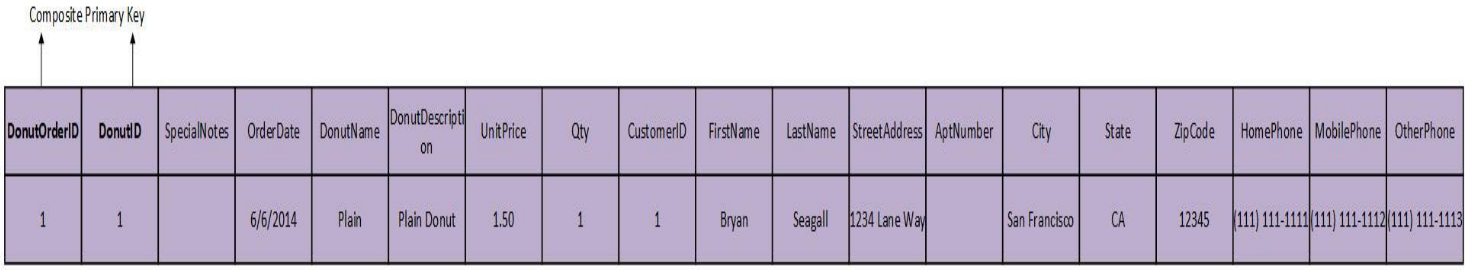
Darryl Lardizabal

Data Management – Applications – C170

January 25, 2018

A. Construct a normalized model to represent the donut shop smartphone application database that supports the scenario above by doing the following:

1NF



Composite Primary Key

DonutOrderID	DonutID	SpecialNotes	OrderDate	DonutName	DonutDescription	UnitPrice	Qty	CustomerID	FirstName	LastName	StreetAddress	AptNumber	City	State	ZipCode	HomePhone	MobilePhone	OtherPhone
1	1		6/6/2014	Plain	Plain Donut	1.50	1	1	Bryan	Seagall	1234 Lane Way		San Francisco	CA	12345	(111) 111-1111	(111) 111-1112	(111) 111-1113

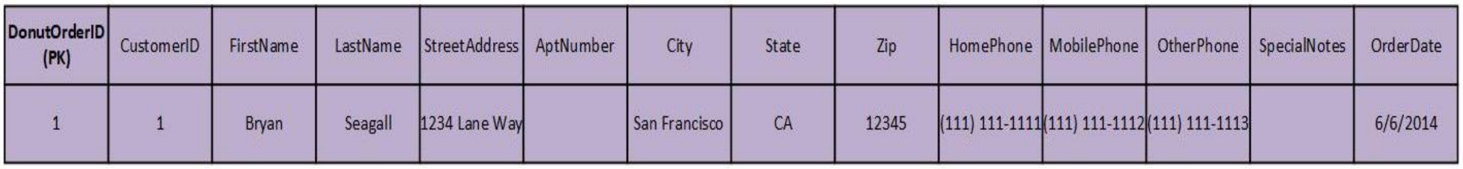
The goal with the 1NF Table is fairly straight forward:

- Determine a primary key.
- The values should be atomic.
- Make sure there are no repeating groups.

I chose donut order ID and donut ID as a composite primary because it seemed to provide a unique value for every record that could be in this table. I wanted to make sure I didn't use order date because there could be multiple orders on the same date. I also stored each repeating group in its own record; and broke up the multivalued field of city, state, zip into separate fields to make sure each is atomic.

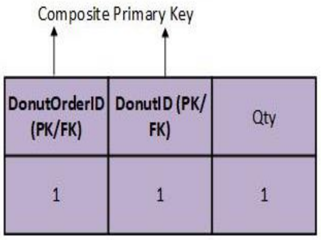
2NF

Invoice



DonutOrderID (PK)	CustomerID	FirstName	LastName	StreetAddress	AptNumber	City	State	Zip	HomePhone	MobilePhone	OtherPhone	SpecialNotes	OrderDate
1	1	Bryan	Seagall	1234 Lane Way		San Francisco	CA	12345	(111) 111-1111	(111) 111-1112	(111) 111-1113		6/6/2014

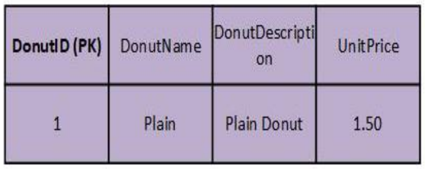
Invoice Line Item



Composite Primary Key

DonutOrderID (PK/FK)	DonutID (PK/FK)	Qty
1	1	1

Donut

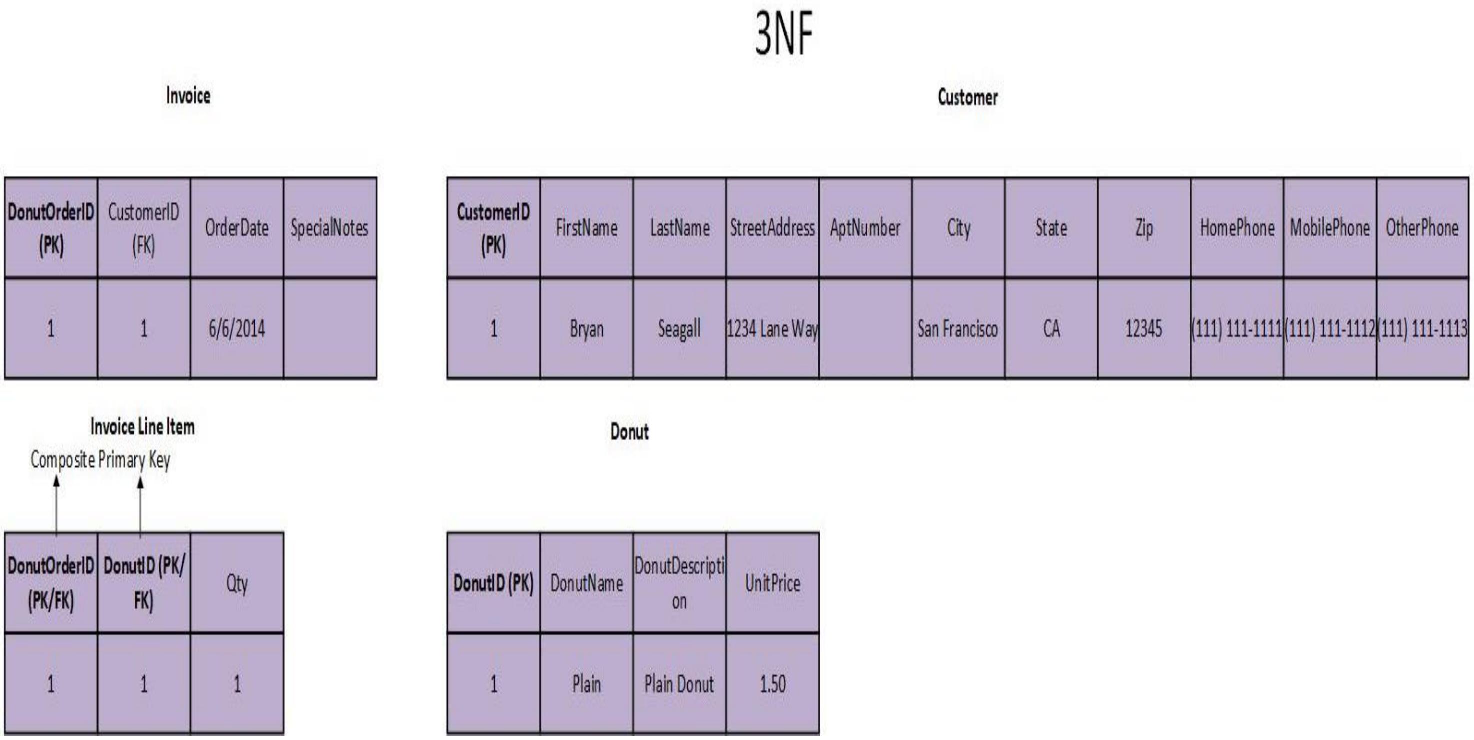


DonutID (PK)	DonutName	DonutDescription	UnitPrice
1	Plain	Plain Donut	1.50

The goal with 2NF Tables is:

- The table should meet all the requirements for the first normal form.
- All functional dependencies should be removed from the tables.
- Each of the tables should have all primary and/or foreign keys designated.

The first goal was since I had a composite primary key for my 1NF, I wanted to check for partial dependencies. I split up The original 1NF table into a table for Invoice, Donut, and Invoice Line Item. Anything that involved donut ID (Primary Key) 1 would involve the same donut name, donut description, and unit price and you didn't need donut order ID to determine those pieces of data. Qty is the only piece in this data set that needed both donut order ID (Primary Key/Foreign Key – pulled from Invoice table) and donut ID (Primary Key/Foreign Key – pulled from Donut Table) to determine the quantity purchased. The rest of the information was what was left after I pulled the information out for the Donut table and the Invoice Line Item table and put into the Invoice Table, which was partially depedent on donut order ID.



3NF

Invoice

DonutOrderID (PK)	CustomerID (FK)	OrderDate	SpecialNotes
1	1	6/6/2014	

Customer

CustomerID (PK)	FirstName	LastName	StreetAddress	AptNumber	City	State	Zip	HomePhone	MobilePhone	OtherPhone
1	Bryan	Seagall	1234 Lane Way		San Francisco	CA	12345	(111) 111-1111	(111) 111-1112	(111) 111-1113

Invoice Line Item

Composite Primary Key

DonutOrderID (PK/FK)	**DonutID (PK/ FK)**	Qty
1	1	1

Donut

DonutID (PK)	DonutName	DonutDescription	UnitPrice
1	Plain	Plain Donut	1.50

The goal with the 3NF Tables is:

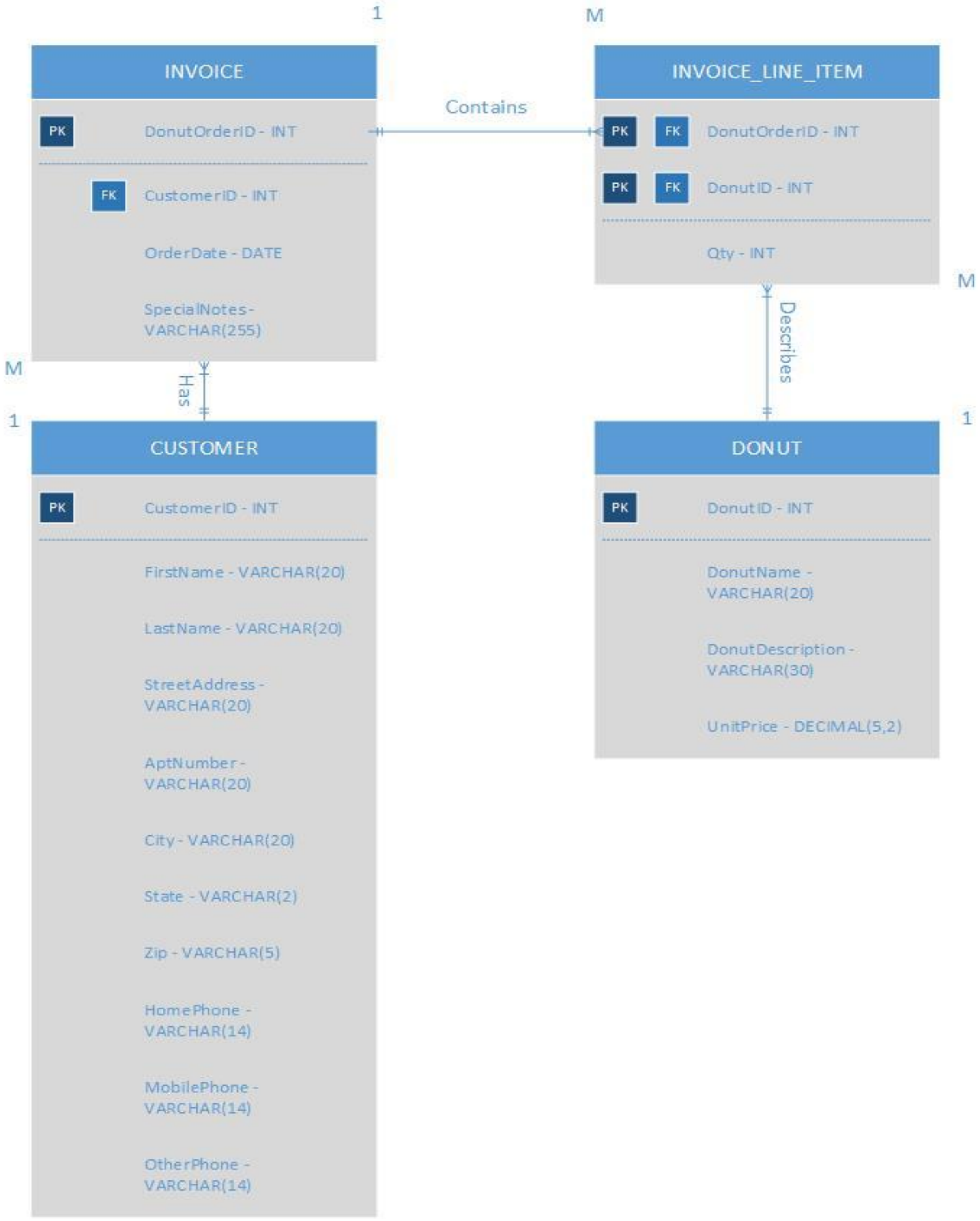
- The tables should meet all the requirements for the first and second normal forms.
- All transitive dependencies should be removed from the tables.
- Each of the tables should have all primary and/or foreign keys designated.

Here in the 3NF form, Customer table was created because whenever you have the Customer ID it will always have the same first name, last name, street address, apt number, city, state, zip, home phone, mobile phone, and other phone. Thus, the invoice table was now split into two tables Invoice and Customer.

All tables now meet the requirements for 3NF.

B. Create an entity-relationship (E-R) diagram, using the tables you designed in third normal form from part A1C…

Darryl Lardizabal
C170 Requirements B
January 16, 2018



A. The entities that I selected in the diagram are: Invoice, Invoice_Line_Item, Customer, and Donut. Entity names are normally chosen as single words that are easily recognizable.

- The Invoice entity was selected as it seemed to represent the order information.
- The Invoice_Line_Item entity was selected as it seemed to represent quantity.
- The Customer entity was selected because it matches the attributes that are specific to a customer from the sales order form.
- The Donut entity was selected because it matches the attributes for the donut from the sales order form.

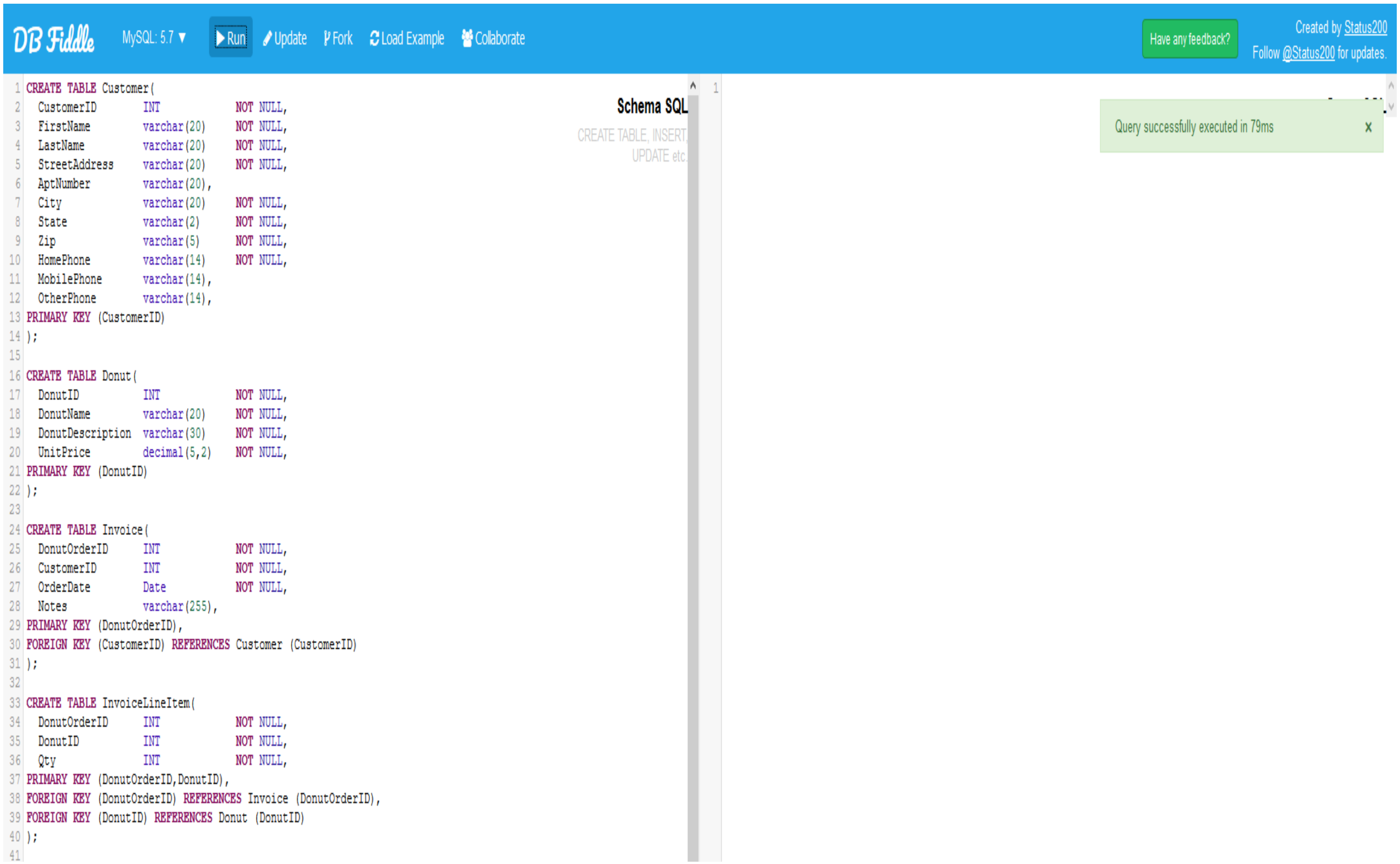
B. Determining the relationships between these entities was rather simple.

- Customers create invoices.
- Invoices have line items.
- Invoice line items describes donut.

C. The cardinality in the diagram is as follows:

- One donut can describe one or many invoice line items. One invoice line item describes only one donut.
- One customer can have one or many invoices. One invoice has only one customer.
- One invoice can contain one or many line items. One invoice line item can have only one invoice.

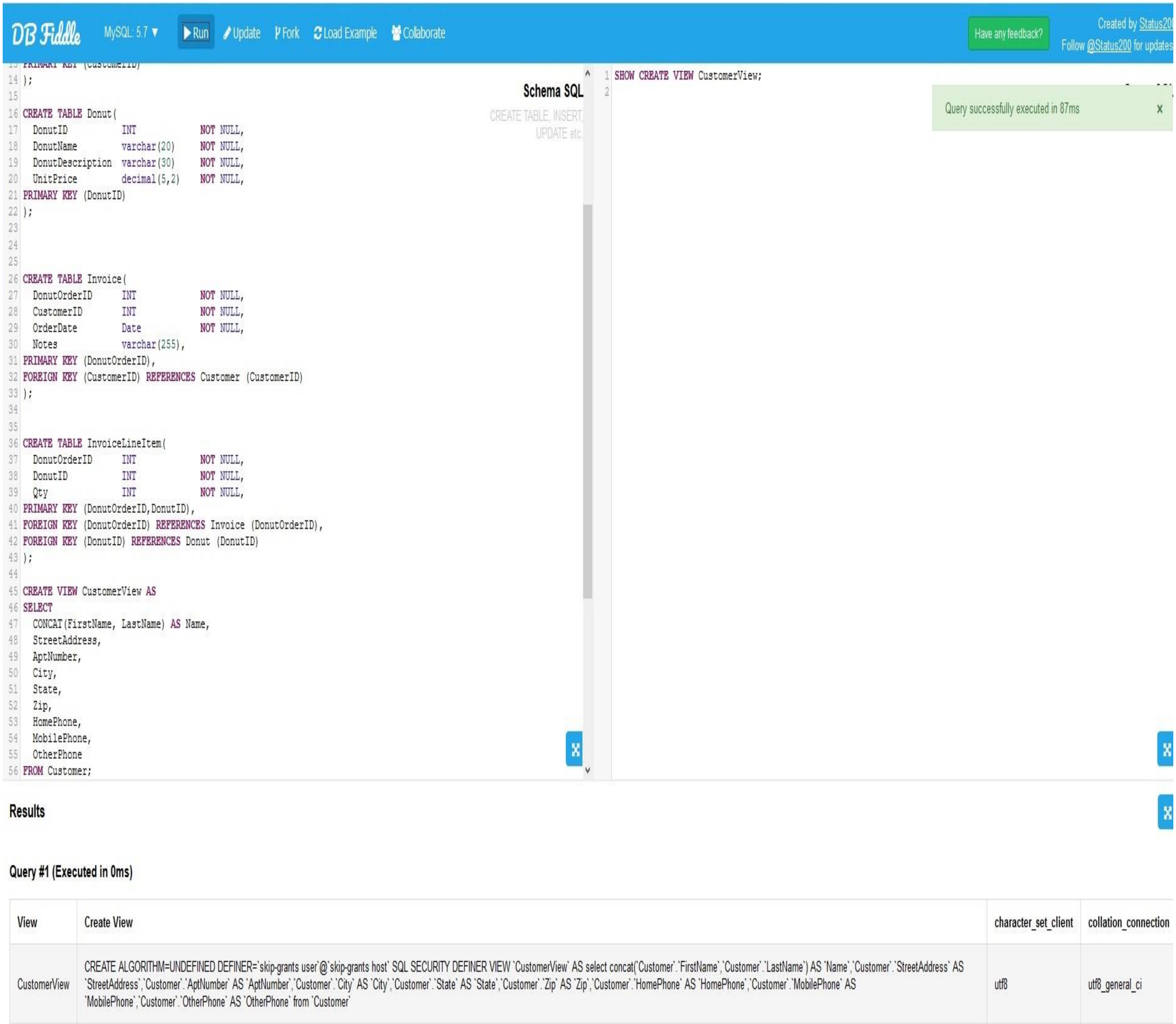
C. Develop the SQL Code to create each of the third normal form tables you designed in part A and refined in part B by doing the following:



```sql
CREATE TABLE Customer(
  CustomerID       INT            NOT NULL,
  FirstName        varchar(20)    NOT NULL,
  LastName         varchar(20)    NOT NULL,
  StreetAddress    varchar(20)    NOT NULL,
  AptNumber        varchar(20),
  City             varchar(20)    NOT NULL,
  State            varchar(2)     NOT NULL,
  Zip              varchar(5)     NOT NULL,
  HomePhone        varchar(14)    NOT NULL,
  MobilePhone      varchar(14),
  OtherPhone       varchar(14),
PRIMARY KEY (CustomerID)
);

CREATE TABLE Donut(
  DonutID          INT            NOT NULL,
  DonutName        varchar(20)    NOT NULL,
  DonutDescription varchar(30)    NOT NULL,
  UnitPrice        decimal(5,2)   NOT NULL,
PRIMARY KEY (DonutID)
);

CREATE TABLE Invoice(
  DonutOrderID     INT            NOT NULL,
  CustomerID       INT            NOT NULL,
  OrderDate        Date           NOT NULL,
  Notes            varchar(255),
PRIMARY KEY (DonutOrderID),
FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);

CREATE TABLE InvoiceLineItem(
  DonutOrderID     INT            NOT NULL,
  DonutID          INT            NOT NULL,
  Qty              INT            NOT NULL,
PRIMARY KEY (DonutOrderID,DonutID),
FOREIGN KEY (DonutOrderID) REFERENCES Invoice (DonutOrderID),
FOREIGN KEY (DonutID) REFERENCES Donut (DonutID)
);
```

Query successfully executed in 79ms

D. Develop the SQL Code to create a “View”:



```
PRIMARY KEY (CustomerID)
);

CREATE TABLE Donut(
  DonutID          INT            NOT NULL,
  DonutName        varchar(20)    NOT NULL,
  DonutDescription varchar(30)    NOT NULL,
  UnitPrice        decimal(5,2)   NOT NULL,
PRIMARY KEY (DonutID)
);



CREATE TABLE Invoice(
  DonutOrderID     INT            NOT NULL,
  CustomerID       INT            NOT NULL,
  OrderDate        Date           NOT NULL,
  Notes            varchar(255),
PRIMARY KEY (DonutOrderID),
FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);


CREATE TABLE InvoiceLineItem(
  DonutOrderID     INT            NOT NULL,
  DonutID          INT            NOT NULL,
  Qty              INT            NOT NULL,
PRIMARY KEY (DonutOrderID,DonutID),
FOREIGN KEY (DonutOrderID) REFERENCES Invoice (DonutOrderID),
FOREIGN KEY (DonutID) REFERENCES Donut (DonutID)
);

CREATE VIEW CustomerView AS
SELECT
  CONCAT(FirstName, LastName) AS Name,
  StreetAddress,
  AptNumber,
  City,
  State,
  Zip,
  HomePhone,
  MobilePhone,
  OtherPhone
FROM Customer;
```

```
SHOW CREATE VIEW CustomerView;
```

Query successfully executed in 87ms

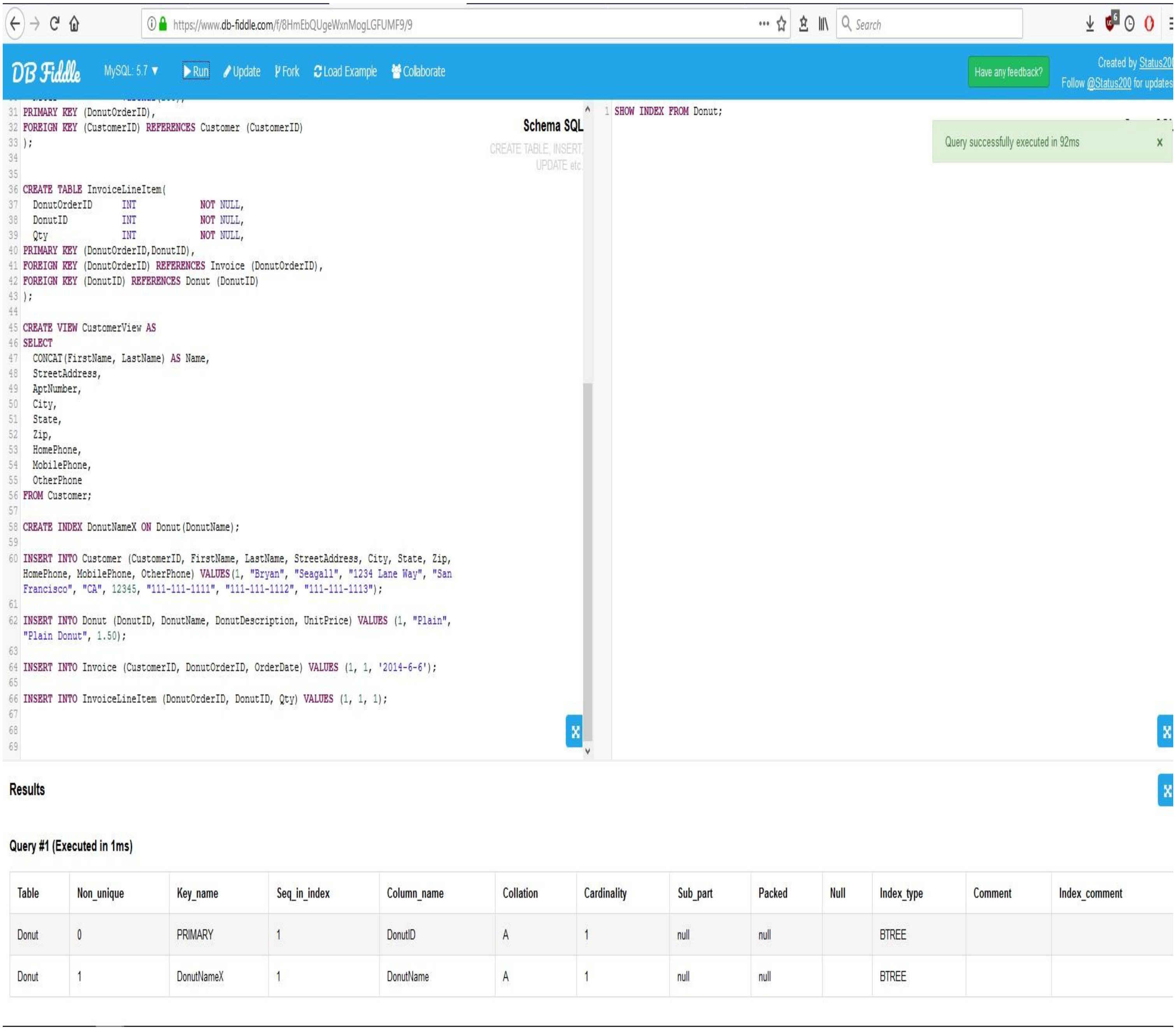
x

Results

Query #1 (Executed in 0ms)

View	Create View	character_set_client	collation_connection
CustomerView	CREATE ALGORITHM=UNDEFINED DEFINER=`skip-grants user`@`skip-grants host` SQL SECURITY DEFINER VIEW `CustomerView` AS select concat(`Customer`.`FirstName`,`Customer`.`LastName`) AS `Name`,`Customer`.`StreetAddress` AS `StreetAddress`,`Customer`.`AptNumber` AS `AptNumber`,`Customer`.`City` AS `City`,`Customer`.`State` AS `State`,`Customer`.`Zip` AS `Zip`,`Customer`.`HomePhone` AS `HomePhone`,`Customer`.`MobilePhone` AS `MobilePhone`,`Customer`.`OtherPhone` AS `OtherPhone` from `Customer`	utf8	utf8_general_ci

E. Develop the SQL Code to create an “Index”:


https://www.db-fiddle.com/f/8HmEbQUgeWxnMogLGFUMF9/9
Search
DB Fiddle
MySQL: 5.7 ▼
▶ Run
Update
Fork
Load Example
Collaborate
Have any feedback?
Created by Status200
Follow @Status200 for updates
Schema SQL
CREATE TABLE, INSERT, UPDATE etc.

```sql
PRIMARY KEY (DonutOrderID),
FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);


CREATE TABLE InvoiceLineItem(
  DonutOrderID      INT           NOT NULL,
  DonutID           INT           NOT NULL,
  Qty               INT           NOT NULL,
PRIMARY KEY (DonutOrderID,DonutID),
FOREIGN KEY (DonutOrderID) REFERENCES Invoice (DonutOrderID),
FOREIGN KEY (DonutID) REFERENCES Donut (DonutID)
);

CREATE VIEW CustomerView AS
SELECT
  CONCAT(FirstName, LastName) AS Name,
  StreetAddress,
  AptNumber,
  City,
  State,
  Zip,
  HomePhone,
  MobilePhone,
  OtherPhone
FROM Customer;

CREATE INDEX DonutNameX ON Donut(DonutName);

INSERT INTO Customer (CustomerID, FirstName, LastName, StreetAddress, City, State, Zip,
HomePhone, MobilePhone, OtherPhone) VALUES(1, "Bryan", "Seagall", "1234 Lane Way", "San
Francisco", "CA", 12345, "111-111-1111", "111-111-1112", "111-111-1113");

INSERT INTO Donut (DonutID, DonutName, DonutDescription, UnitPrice) VALUES (1, "Plain",
"Plain Donut", 1.50);

INSERT INTO Invoice (CustomerID, DonutOrderID, OrderDate) VALUES (1, 1, '2014-6-6');

INSERT INTO InvoiceLineItem (DonutOrderID, DonutID, Qty) VALUES (1, 1, 1);
```

```sql
SHOW INDEX FROM Donut;
```

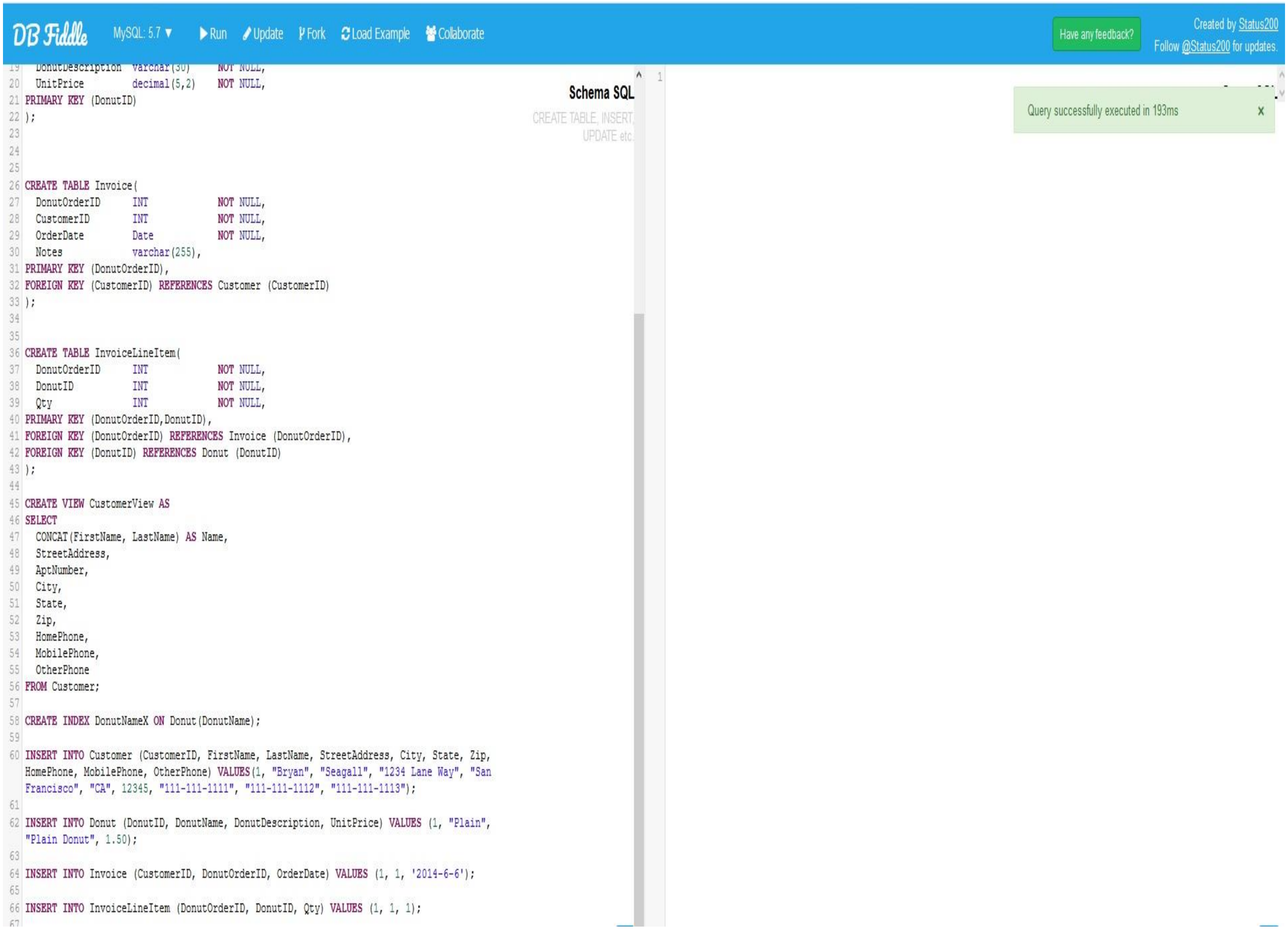
Query successfully executed in 92ms

Results

Query #1 (Executed in 1ms)

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
Donut	0	PRIMARY	1	DonutID	A	1	null	null		BTREE		
Donut	1	DonutNameX	1	DonutName	A	1	null	null		BTREE		

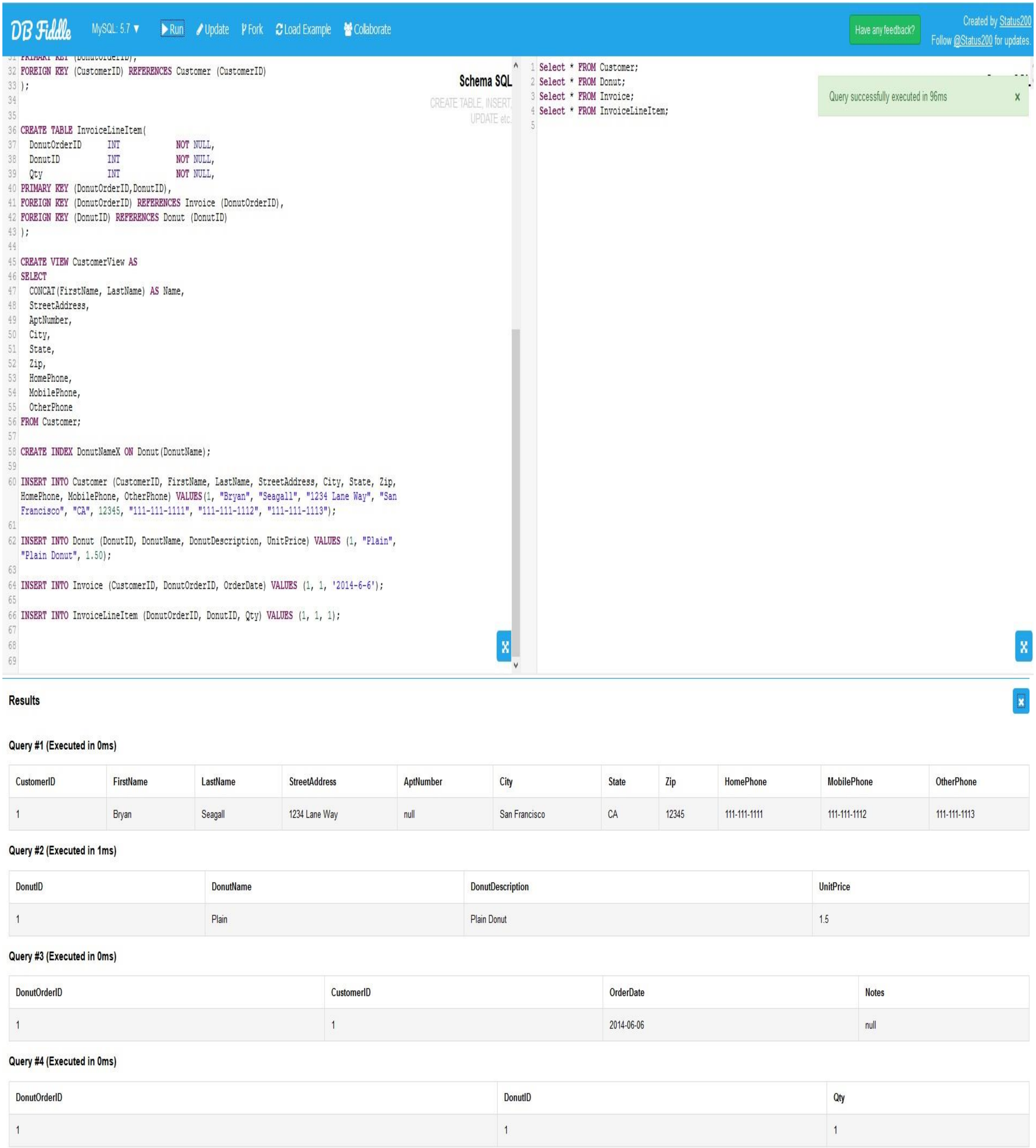
F. Develop the SQL Code to populate all of the tables:



```
  DonutDescription  varchar(30)     NOT NULL,
  UnitPrice         decimal(5,2)    NOT NULL,
PRIMARY KEY (DonutID)
);



CREATE TABLE Invoice(
  DonutOrderID      INT             NOT NULL,
  CustomerID        INT             NOT NULL,
  OrderDate         Date            NOT NULL,
  Notes             varchar(255),
PRIMARY KEY (DonutOrderID),
FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);


CREATE TABLE InvoiceLineItem(
  DonutOrderID      INT             NOT NULL,
  DonutID           INT             NOT NULL,
  Qty               INT             NOT NULL,
PRIMARY KEY (DonutOrderID,DonutID),
FOREIGN KEY (DonutOrderID) REFERENCES Invoice (DonutOrderID),
FOREIGN KEY (DonutID) REFERENCES Donut (DonutID)
);

CREATE VIEW CustomerView AS
SELECT
  CONCAT(FirstName, LastName) AS Name,
  StreetAddress,
  AptNumber,
  City,
  State,
  Zip,
  HomePhone,
  MobilePhone,
  OtherPhone
FROM Customer;

CREATE INDEX DonutNameX ON Donut(DonutName);

INSERT INTO Customer (CustomerID, FirstName, LastName, StreetAddress, City, State, Zip,
HomePhone, MobilePhone, OtherPhone) VALUES(1, "Bryan", "Seagall", "1234 Lane Way", "San
Francisco", "CA", 12345, "111-111-1111", "111-111-1112", "111-111-1113");

INSERT INTO Donut (DonutID, DonutName, DonutDescription, UnitPrice) VALUES (1, "Plain",
"Plain Donut", 1.50);

INSERT INTO Invoice (CustomerID, DonutOrderID, OrderDate) VALUES (1, 1, '2014-6-6');

INSERT INTO InvoiceLineItem (DonutOrderID, DonutID, Qty) VALUES (1, 1, 1);
```

Query successfully executed in 193ms

G. Develop the SQL code to Display the values in a requested table or tables using the tables populated in part F:



```sql
PRIMARY KEY (DonutOrderID),
FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);


CREATE TABLE InvoiceLineItem(
  DonutOrderID      INT            NOT NULL,
  DonutID           INT            NOT NULL,
  Qty               INT            NOT NULL,
PRIMARY KEY (DonutOrderID,DonutID),
FOREIGN KEY (DonutOrderID) REFERENCES Invoice (DonutOrderID),
FOREIGN KEY (DonutID) REFERENCES Donut (DonutID)
);

CREATE VIEW CustomerView AS
SELECT
  CONCAT(FirstName, LastName) AS Name,
  StreetAddress,
  AptNumber,
  City,
  State,
  Zip,
  HomePhone,
  MobilePhone,
  OtherPhone
FROM Customer;

CREATE INDEX DonutNameX ON Donut(DonutName);

INSERT INTO Customer (CustomerID, FirstName, LastName, StreetAddress, City, State, Zip,
HomePhone, MobilePhone, OtherPhone) VALUES(1, "Bryan", "Seagall", "1234 Lane Way", "San
Francisco", "CA", 12345, "111-111-1111", "111-111-1112", "111-111-1113");

INSERT INTO Donut (DonutID, DonutName, DonutDescription, UnitPrice) VALUES (1, "Plain",
"Plain Donut", 1.50);

INSERT INTO Invoice (CustomerID, DonutOrderID, OrderDate) VALUES (1, 1, '2014-6-6');

INSERT INTO InvoiceLineItem (DonutOrderID, DonutID, Qty) VALUES (1, 1, 1);
```

```sql
Select * FROM Customer;
Select * FROM Donut;
Select * FROM Invoice;
Select * FROM InvoiceLineItem;
```

Query successfully executed in 96ms x

Results

Query #1 (Executed in 0ms)

CustomerID	FirstName	LastName	StreetAddress	AptNumber	City	State	Zip	HomePhone	MobilePhone	OtherPhone
1	Bryan	Seagall	1234 Lane Way	null	San Francisco	CA	12345	111-111-1111	111-111-1112	111-111-1113

Query #2 (Executed in 1ms)

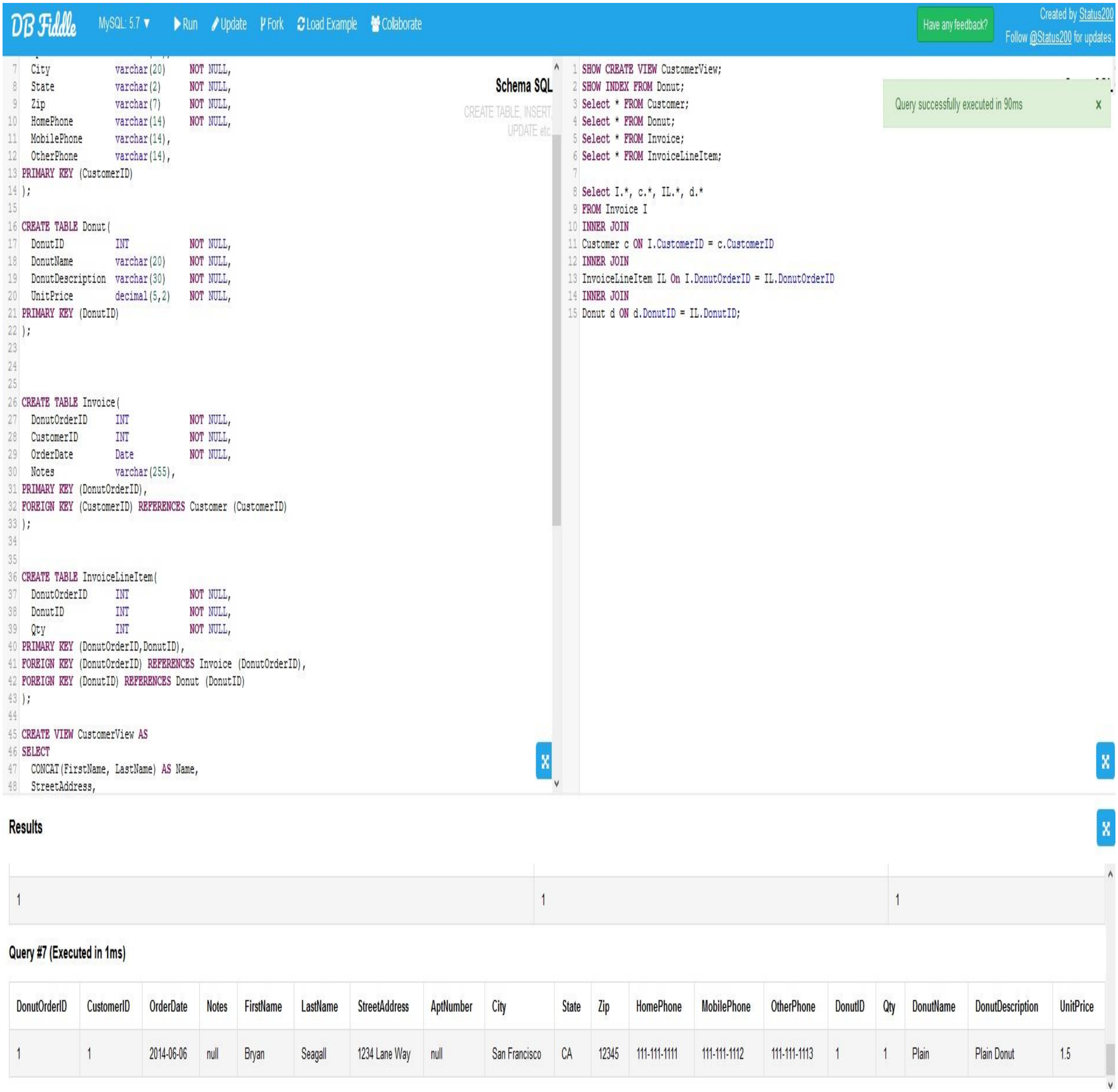
DonutID	DonutName	DonutDescription	UnitPrice
1	Plain	Plain Donut	1.5

Query #3 (Executed in 0ms)

DonutOrderID	CustomerID	OrderDate	Notes
1	1	2014-06-06	null

Query #4 (Executed in 0ms)

DonutOrderID	DonutID	Qty
1	1	1



```sql
  City           varchar(20)     NOT NULL,
  State          varchar(2)      NOT NULL,
  Zip            varchar(7)      NOT NULL,
  HomePhone      varchar(14)     NOT NULL,
  MobilePhone    varchar(14),
  OtherPhone     varchar(14),
PRIMARY KEY (CustomerID)
);

CREATE TABLE Donut(
  DonutID          INT            NOT NULL,
  DonutName        varchar(20)    NOT NULL,
  DonutDescription varchar(30)    NOT NULL,
  UnitPrice        decimal(5,2)   NOT NULL,
PRIMARY KEY (DonutID)
);



CREATE TABLE Invoice(
  DonutOrderID     INT            NOT NULL,
  CustomerID       INT            NOT NULL,
  OrderDate        Date           NOT NULL,
  Notes            varchar(255),
PRIMARY KEY (DonutOrderID),
FOREIGN KEY (CustomerID) REFERENCES Customer (CustomerID)
);


CREATE TABLE InvoiceLineItem(
  DonutOrderID     INT            NOT NULL,
  DonutID          INT            NOT NULL,
  Qty              INT            NOT NULL,
PRIMARY KEY (DonutOrderID,DonutID),
FOREIGN KEY (DonutOrderID) REFERENCES Invoice (DonutOrderID),
FOREIGN KEY (DonutID) REFERENCES Donut (DonutID)
);

CREATE VIEW CustomerView AS
SELECT
  CONCAT(FirstName, LastName) AS Name,
  StreetAddress,
```

```sql
SHOW CREATE VIEW CustomerView;
SHOW INDEX FROM Donut;
Select * FROM Customer;
Select * FROM Donut;
Select * FROM Invoice;
Select * FROM InvoiceLineItem;

Select I.*, c.*, IL.*, d.*
FROM Invoice I
INNER JOIN
Customer c ON I.CustomerID = c.CustomerID
INNER JOIN
InvoiceLineItem IL On I.DonutOrderID = IL.DonutOrderID
INNER JOIN
Donut d ON d.DonutID = IL.DonutID;
```

Query successfully executed in 90ms

x

Results

1	1	1

Query #7 (Executed in 1ms)

DonutOrderID	CustomerID	OrderDate	Notes	FirstName	LastName	StreetAddress	AptNumber	City	State	Zip	HomePhone	MobilePhone	OtherPhone	DonutID	Qty	DonutName	DonutDescription	UnitPrice
1	1	2014-06-06	null	Bryan	Seagall	1234 Lane Way	null	San Francisco	CA	12345	111-111-1111	111-111-1112	111-111-1113	1	1	Plain	Plain Donut	1.5