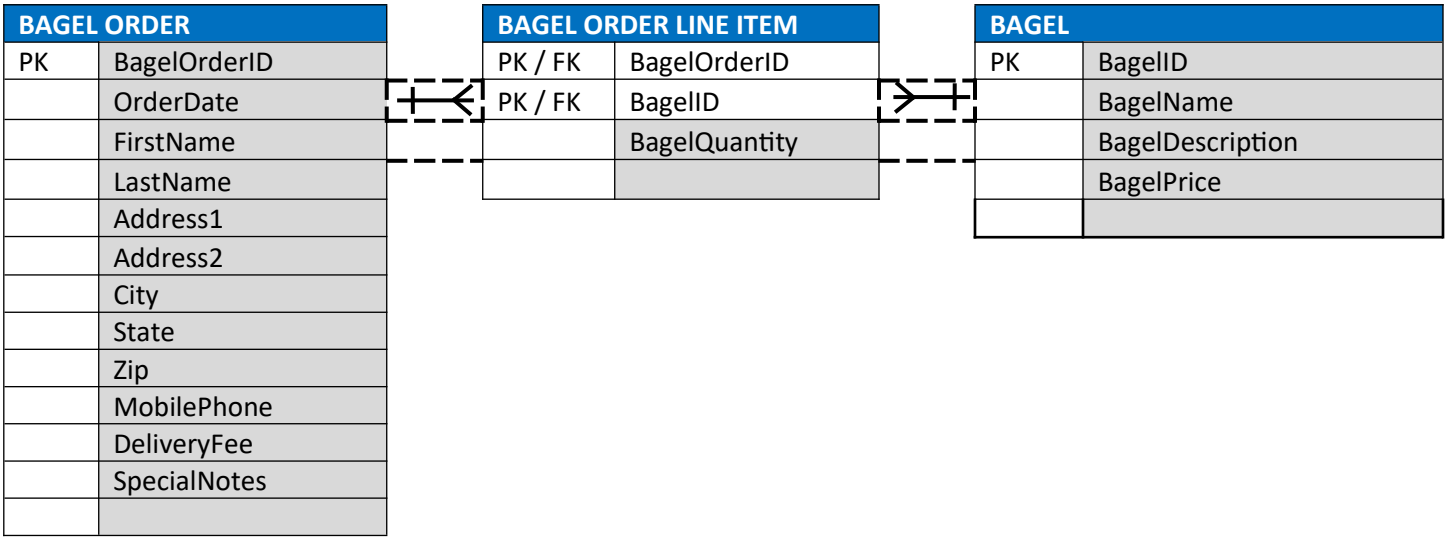
PART A

Nora’s Bagel Bin Database Blueprints

Second Normal Form (2NF)



To achieve 2NF, the BagelOrder table contains information about the order, such as the order date, customer details, delivery fee, and special notes. The BagelOrderLineItem table contains information about each bagel ordered, including the bagel ID and quantity. The Bagel table contains information about each type of bagel, such as the name, description, and price.

BagelOrder to BagelOrderLineItem: This is a one-to-many (1:M) relationship, since each order can have multiple line items, but each line item is associated with only one order.

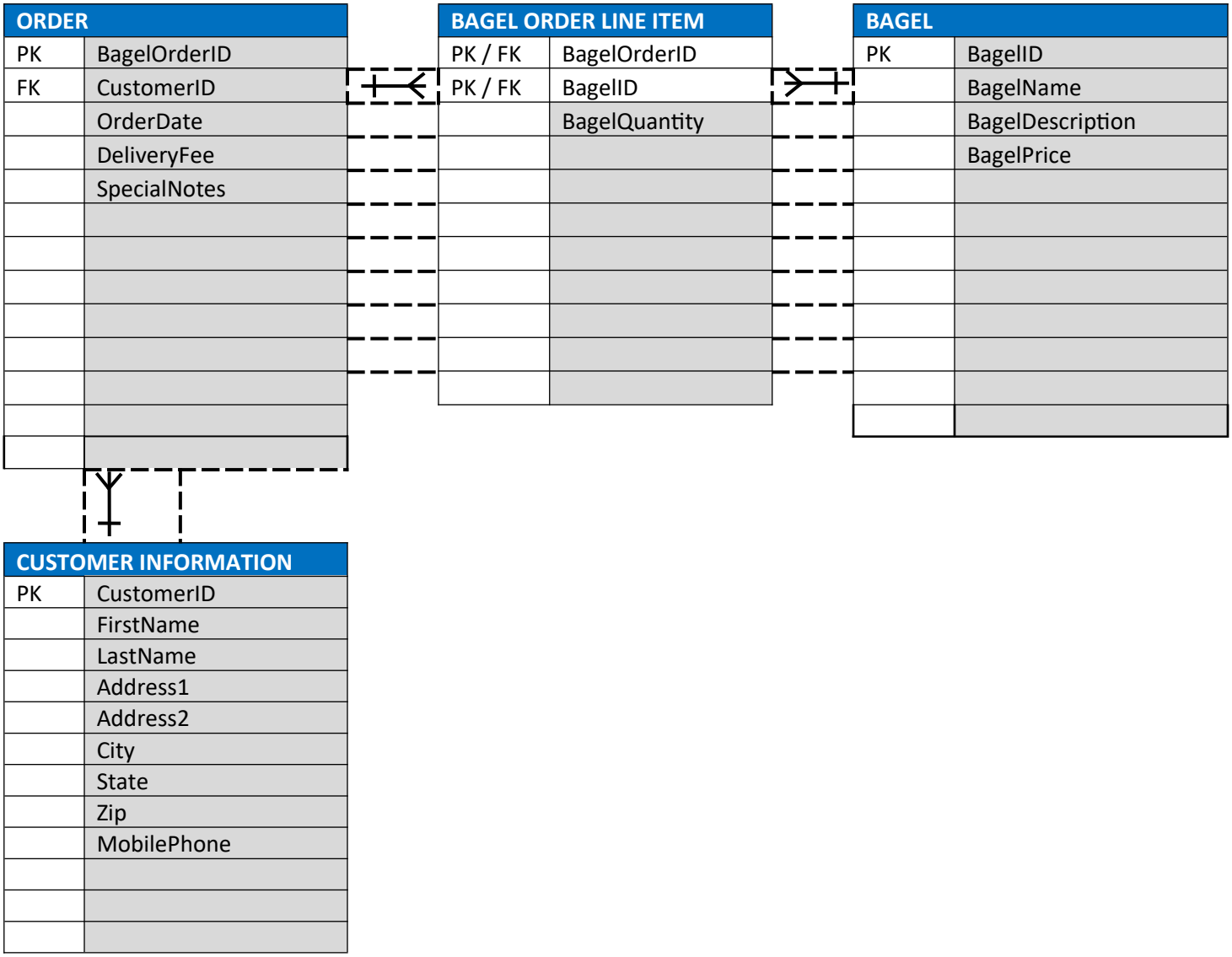
BagelOrderLineItem to Bagel: This is a many-to-one (M:1) relationship, since multiple line items can be associated with a single type of bagel, but each line item can only be associated with one type of bagel.

Bagel to BagelOrderLineItem: This is also a one-to-many (1:M) relationship, since each type of bagel can be included in multiple line items in different orders, but each line item can only specify one type of bagel.

Breaking down the original 1NF table into multiple tables achieves 2NF by removing repeated groups of data and ensuring that each column in each table is dependent on the primary key of that table. In this blueprint, the BagelOrderLineItem table has a composite primary key consisting of the BagelOrderID and BagelID columns, which ensures that each line item is uniquely identified by the order it belongs to and the bagel it contains. The foreign key constraints ensure referential integrity between the tables.

Nora’s Bagel Bin Database Blueprints

Third Normal Form (3NF)



In this blueprint, the Bagel table contains information about each type of bagel, including the bagel ID, bagel name, bagel description, and bagel price. The CustomerInformation table contains information about each customer, including the customer ID, first name, last name, address, city, state, zip code, and mobile phone number. The Order table contains information about each order, including the order ID, the order date, the customer ID, the delivery fee, and any special notes. The BagelOrderLineItem table connects the Order and Bagel tables and contains information about each line item in an order, including the line item ID, the order ID, the bagel ID, the bagel quantity, and any special notes.

The cardinality relationships remain the same as described in the previous 2NF blueprint. The Order table has a many-to-one (M:1) relationship with the CustomerInformation table, as each order is associated with one customer, but each customer can have many orders. The OrderLineItem table has a many-to-one (M:1) relationship with both the Order and Bagel tables. Finally, the Bagel table has a one-to-many (1:M) relationship with the OrderLineItem table, since each type of bagel can be included in multiple line items in different orders.

In this design, I have achieved third normal form (3NF) by removing any transitive dependencies that existed between the attributes in the original BagelOrder table.

In the original BagelOrder table, we had the following attributes:

BagelOrderID
BagelID
OrderDate
FirstName
LastName
Address1
Address2
City
State
Zip
MobilePhone
DeliveryFee
BagelName
BagelDescription
BagelPrice
BagelQuantity
SpecialNotes

I identified two transitive dependencies in this table:

The attributes FirstName, LastName, Address1, Address2, City, State, Zip, and MobilePhone depend on the CustomerID, but not on the BagelOrderID.

The attributes BagelName, BagelDescription, and BagelPrice depend on the BagelID, but not on the BagelOrderID.

To remove these transitive dependencies, I split the original BagelOrder table into three tables:

BagelOrder, which includes the attributes that depend only on the BagelOrderID: BagelOrderID, OrderDate, CustomerID, DeliveryFee, and SpecialNotes.

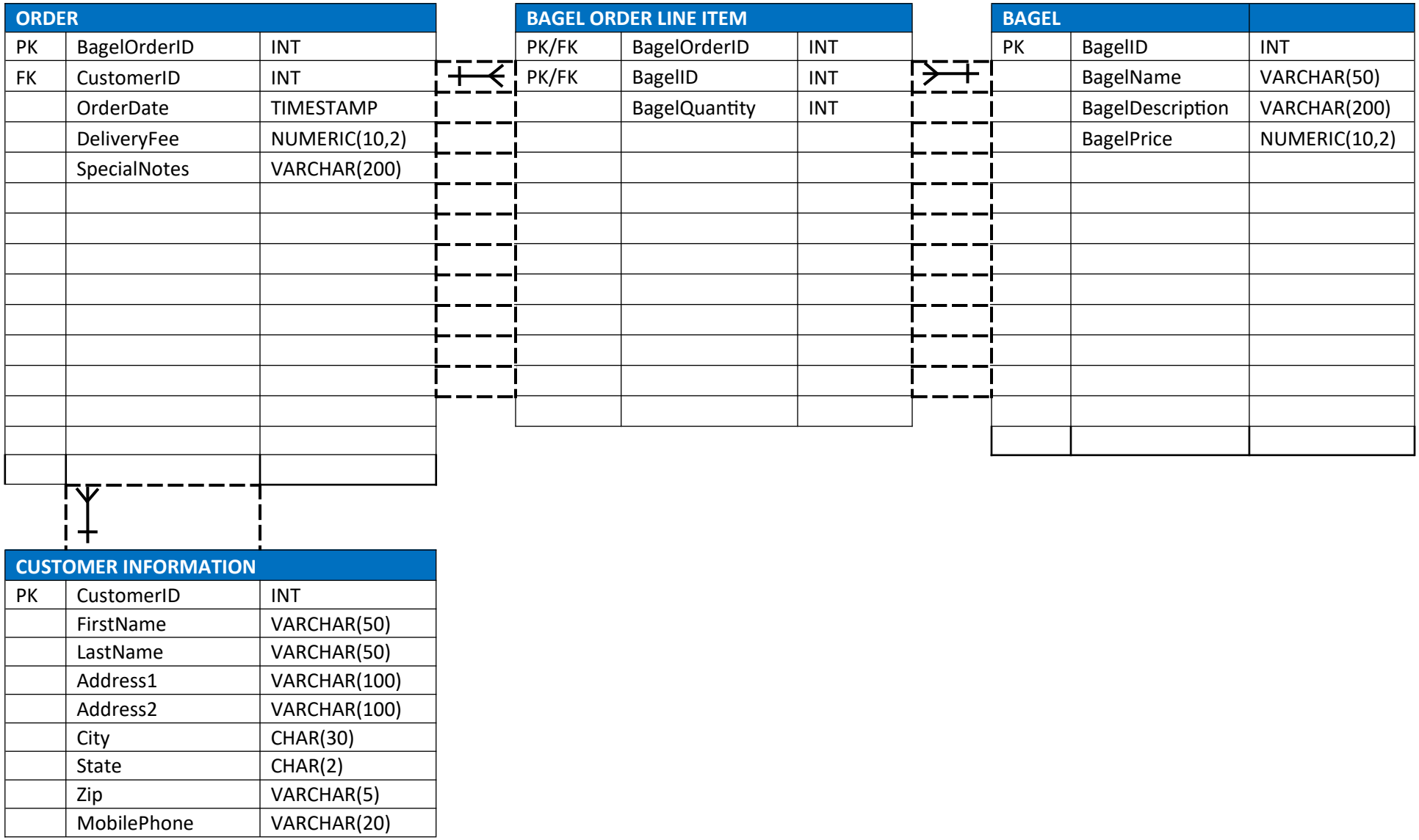
BagelOrderLineItem, which includes the attributes that depend on both the BagelOrderID and the BagelID: BagelOrderLineItemID, BagelOrderID, BagelID, BagelQuantity, and SpecialNotes.

Bagel, which includes the attributes that depend only on the BagelID: BagelID, BagelName, BagelDescription, and BagelPrice.

By doing this, I have eliminated the transitive dependencies in the original BagelOrder table and achieved third normal form (3NF). Each table now contains only attributes that are functionally dependent on its primary key, and I have minimized data redundancy by creating separate tables for related data.

Nora’s Bagel Bin Database Blueprints

Final Physical Database Model



ORDER		
PK	BagelOrderID	INT
FK	CustomerID	INT
	OrderDate	TIMESTAMP
	DeliveryFee	NUMERIC(10,2)
	SpecialNotes	VARCHAR(200)

BAGEL ORDER LINE ITEM		
PK/FK	BagelOrderID	INT
PK/FK	BagelID	INT
	BagelQuantity	INT

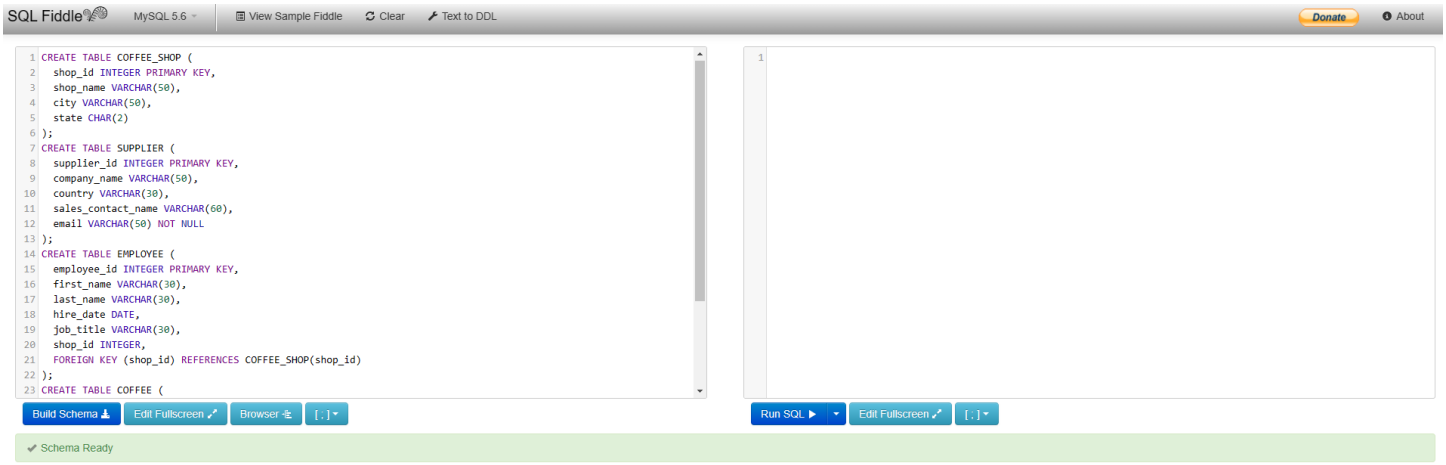
BAGEL		
PK	BagelID	INT
	BagelName	VARCHAR(50)
	BagelDescription	VARCHAR(200)
	BagelPrice	NUMERIC(10,2)

CUSTOMER INFORMATION		
PK	CustomerID	INT
	FirstName	VARCHAR(50)
	LastName	VARCHAR(50)
	Address1	VARCHAR(100)
	Address2	VARCHAR(100)
	City	CHAR(30)
	State	CHAR(2)
	Zip	VARCHAR(5)
	MobilePhone	VARCHAR(20)

PART B

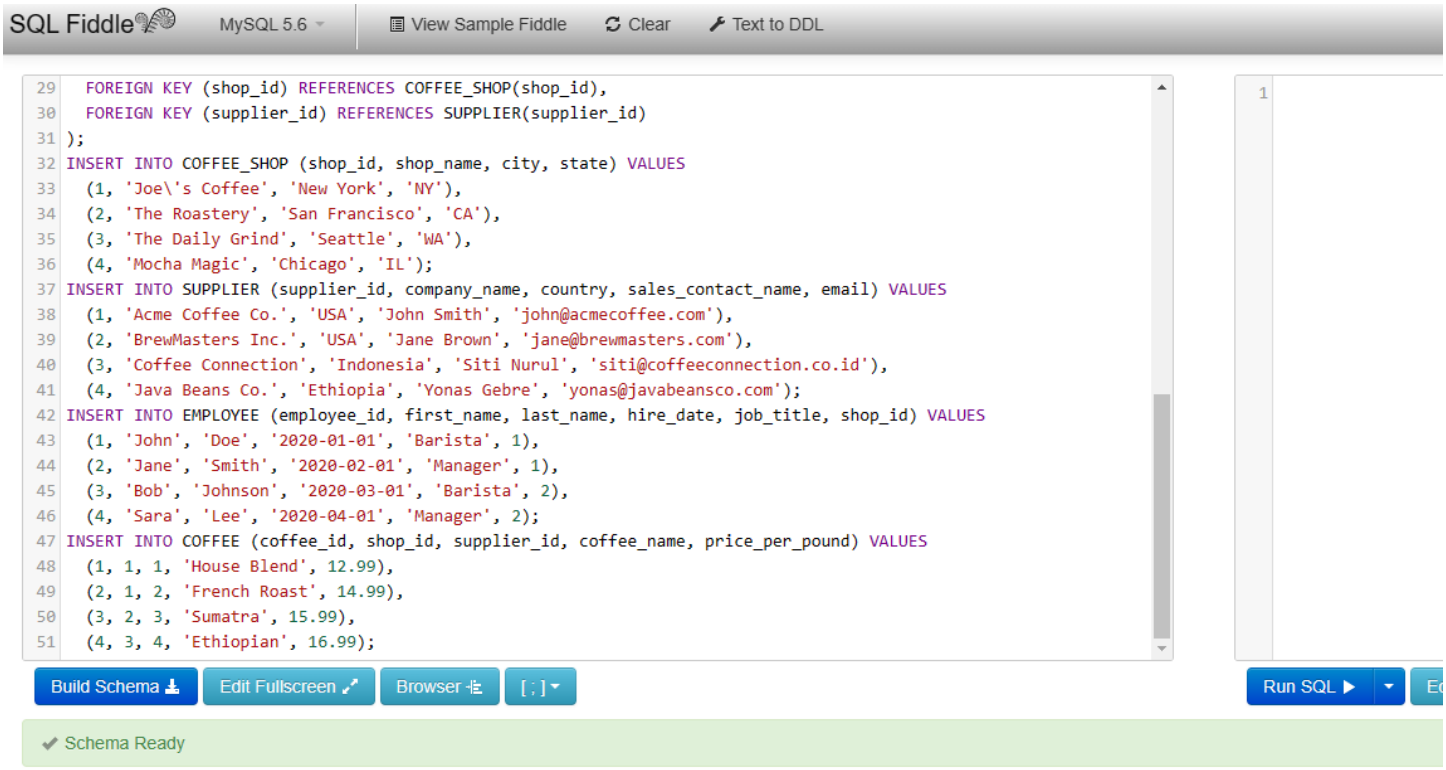
Jaunty Coffee Co. ERD

B-1:




```
CREATE TABLE COFFEE_SHOP (
  shop_id INTEGER PRIMARY KEY,
  shop_name VARCHAR(50),
  city VARCHAR(50),
  state CHAR(2)
);
CREATE TABLE SUPPLIER (
  supplier_id INTEGER PRIMARY KEY,
  company_name VARCHAR(50),
  country VARCHAR(30),
  sales_contact_name VARCHAR(60),
  email VARCHAR(50) NOT NULL
);
CREATE TABLE EMPLOYEE (
  employee_id INTEGER PRIMARY KEY,
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  hire_date DATE,
  job_title VARCHAR(30),
  shop_id INTEGER,
  FOREIGN KEY (shop_id) REFERENCES COFFEE_SHOP(shop_id)
);
CREATE TABLE COFFEE (
  coffee_id INTEGER PRIMARY KEY,
  shop_id INTEGER,
  supplier_id INTEGER,
  coffee_name VARCHAR(30),
  price_per_pound NUMERIC(5,2),
  FOREIGN KEY (shop_id) REFERENCES COFFEE_SHOP(shop_id),
  FOREIGN KEY (supplier_id) REFERENCES SUPPLIER(supplier_id)
);
```

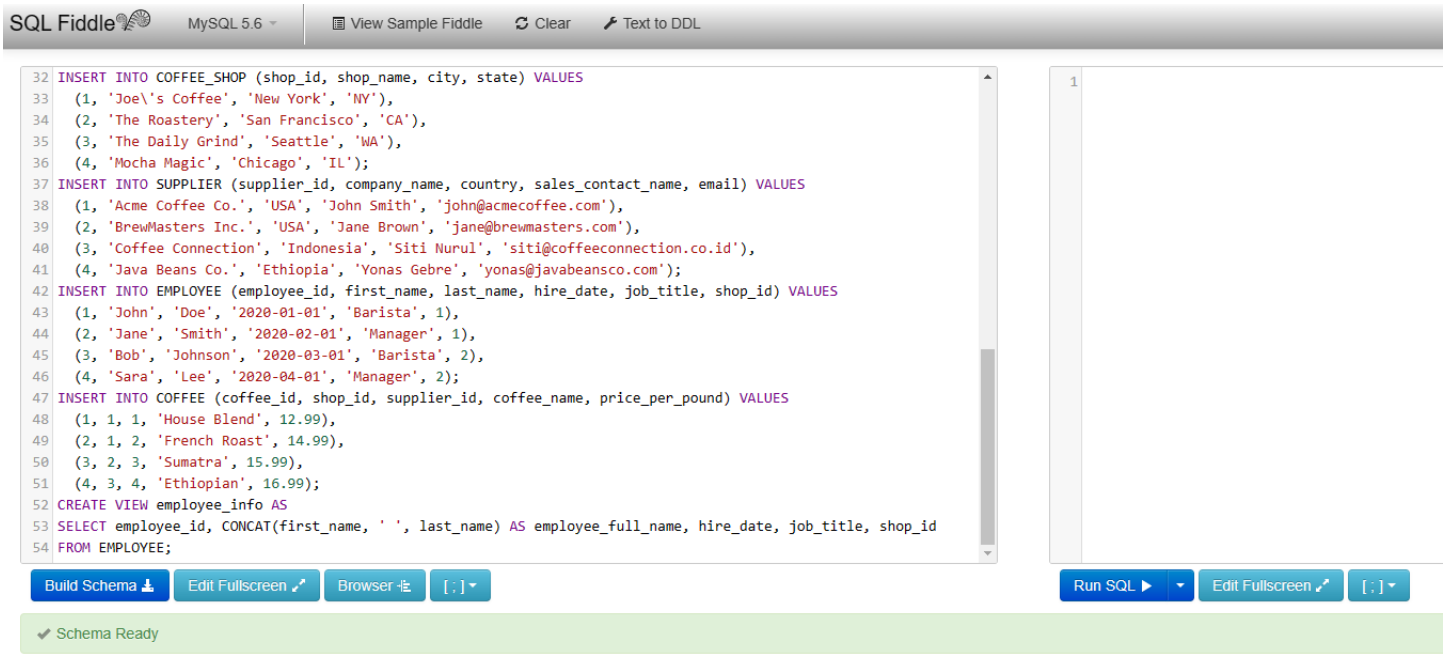
B2:




```
  FOREIGN KEY (shop_id) REFERENCES COFFEE_SHOP(shop_id),
  FOREIGN KEY (supplier_id) REFERENCES SUPPLIER(supplier_id)
);
INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state) VALUES
  (1, 'Joe\'s Coffee', 'New York', 'NY'),
  (2, 'The Roastery', 'San Francisco', 'CA'),
  (3, 'The Daily Grind', 'Seattle', 'WA'),
  (4, 'Mocha Magic', 'Chicago', 'IL');
INSERT INTO SUPPLIER (supplier_id, company_name, country, sales_contact_name, email) VALUES
  (1, 'Acme Coffee Co.', 'USA', 'John Smith', 'john@acmecoffee.com'),
  (2, 'BrewMasters Inc.', 'USA', 'Jane Brown', 'jane@brewmasters.com'),
  (3, 'Coffee Connection', 'Indonesia', 'Siti Nurul', 'siti@coffeeconnection.co.id'),
  (4, 'Java Beans Co.', 'Ethiopia', 'Yonas Gebre', 'yonas@javabeansco.com');
INSERT INTO EMPLOYEE (employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES
  (1, 'John', 'Doe', '2020-01-01', 'Barista', 1),
  (2, 'Jane', 'Smith', '2020-02-01', 'Manager', 1),
  (3, 'Bob', 'Johnson', '2020-03-01', 'Barista', 2),
  (4, 'Sara', 'Lee', '2020-04-01', 'Manager', 2);
INSERT INTO COFFEE (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound) VALUES
  (1, 1, 1, 'House Blend', 12.99),
  (2, 1, 2, 'French Roast', 14.99),
  (3, 2, 3, 'Sumatra', 15.99),
  (4, 3, 4, 'Ethiopian', 16.99);
```

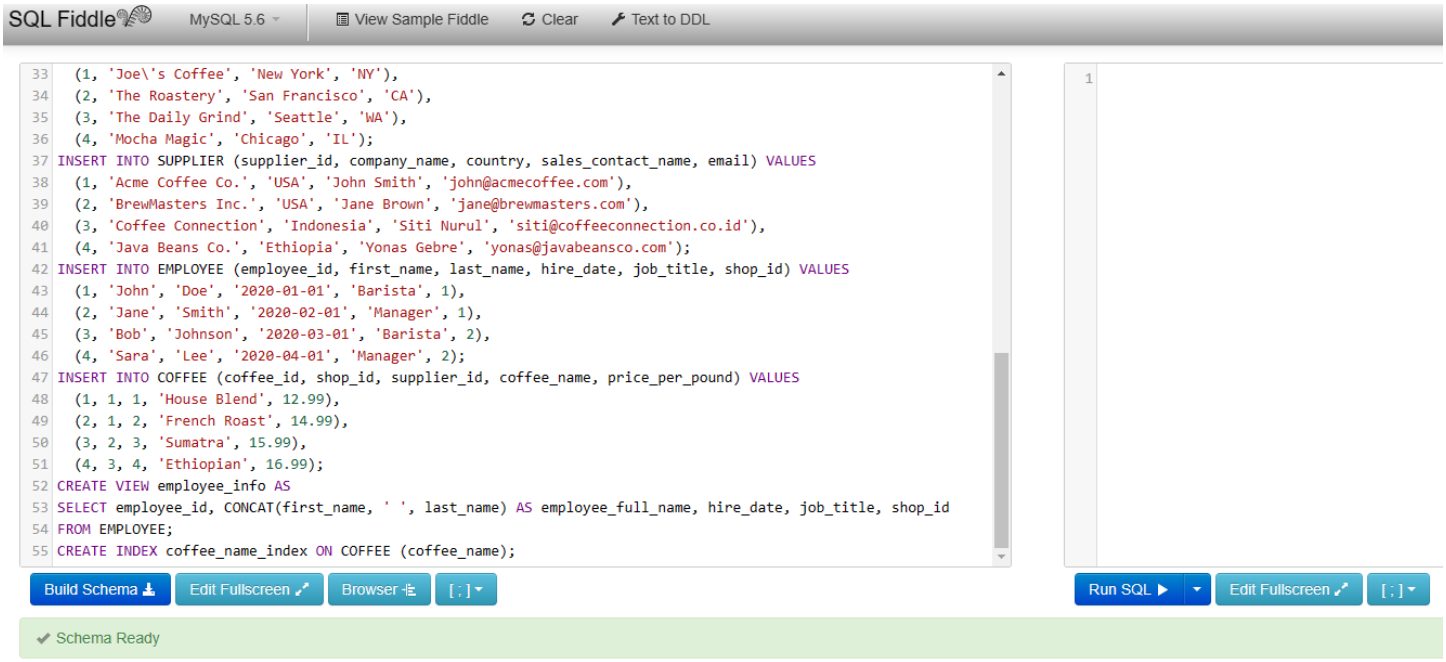

```
INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state) VALUES
  (1, 'Joe\'s Coffee', 'New York', 'NY'),
  (2, 'The Roastery', 'San Francisco', 'CA'),
  (3, 'The Daily Grind', 'Seattle', 'WA'),
  (4, 'Mocha Magic', 'Chicago', 'IL');
INSERT INTO SUPPLIER (supplier_id, company_name, country, sales_contact_name, email) VALUES
  (1, 'Acme Coffee Co.', 'USA', 'John Smith', 'john@acmecoffee.com'),
  (2, 'BrewMasters Inc.', 'USA', 'Jane Brown', 'jane@brewmasters.com'),
  (3, 'Coffee Connection', 'Indonesia', 'Siti Nurul', 'siti@coffeeconnection.co.id'),
  (4, 'Java Beans Co.', 'Ethiopia', 'Yonas Gebre', 'yonas@javabeansco.com');
INSERT INTO EMPLOYEE (employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES
  (1, 'John', 'Doe', '2020-01-01', 'Barista', 1),
  (2, 'Jane', 'Smith', '2020-02-01', 'Manager', 1),
  (3, 'Bob', 'Johnson', '2020-03-01', 'Barista', 2),
  (4, 'Sara', 'Lee', '2020-04-01', 'Manager', 2);
INSERT INTO COFFEE (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound) VALUES
  (1, 1, 1, 'House Blend', 12.99),
  (2, 1, 2, 'French Roast', 14.99),
  (3, 2, 3, 'Sumatra', 15.99),
  (4, 3, 4, 'Ethiopian', 16.99);
```

B3:



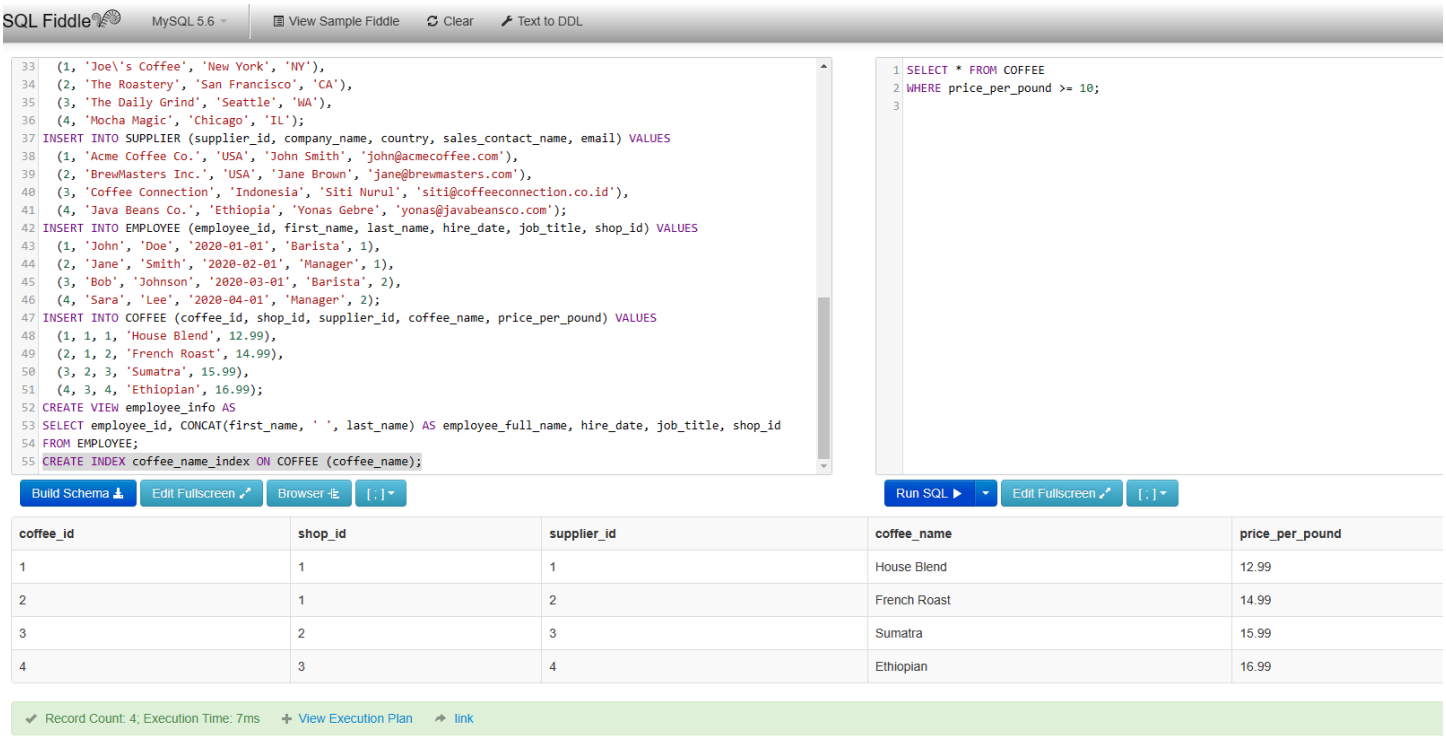
CREATE VIEW employee_info AS
SELECT employee_id, CONCAT(first_name, ' ', last_name) AS employee_full_name, hire_date, job_title, shop_id
FROM EMPLOYEE;

B4:



CREATE INDEX coffee_name_index ON COFFEE (coffee_name);

B5:

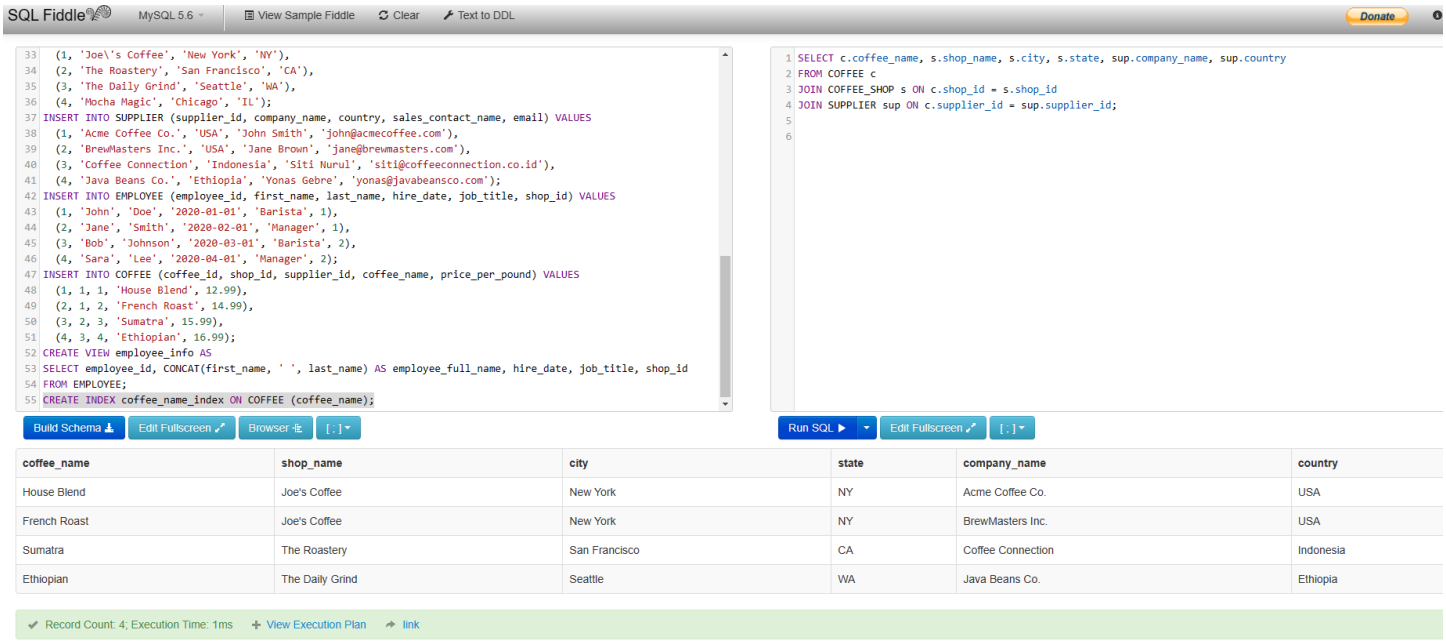


coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
1	1	1	House Blend	12.99
2	1	2	French Roast	14.99
3	2	3	Sumatra	15.99
4	3	4	Ethiopian	16.99

Record Count: 4; Execution Time: 7ms View Execution Plan link

SELECT * FROM COFFEE
WHERE price_per_pound >= 10;

B6:




```
SELECT c.coffee_name, s.shop_name, s.city, s.state, sup.company_name, sup.country
FROM COFFEE c
JOIN COFFEE_SHOP s ON c.shop_id = s.shop_id
JOIN SUPPLIER sup ON c.supplier_id = sup.supplier_id;
```