Task 1: Normalization & Database Design

Xavier Loera Flores

Part A: Nora's Bagel Bin

A1: Second Normal Form

Second Normal Form (2NF)

BAGEL ORDER	
PK	Bagel Order ID
	Order Date
	First Name
	Last Name
	Address 1
	Address 2
	City
	State
	Zip
	Mobile Phone
	Delivery Fee
	Special Notes

1:M

BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Bagel Quantity

M:1

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

Attribute Explanation

The Bagel Order table features the Bagel Order Id, Order Date, First Name, Last Name, Address 1, Address 2, City, State, Zip, Mobile Phone, Delivery Fee, and Special Notes attributes since these attributes can all exclusively depend on the Bagel Order ID.

The Bagel Order Line Item table features the Bagel Order ID, Bagel ID, and Bagel Quantity since the Bagel Quantity is dependent on the unique compound attribute consisting of the Bagel Order ID and the Bagel ID.

The Bagel table features the Bagel ID, Bagel Name, Bagel Description, and Bagel Price attributes since these attributes are dependent on and can be referenced by just the Bagel ID without needing a Bagel Order ID.

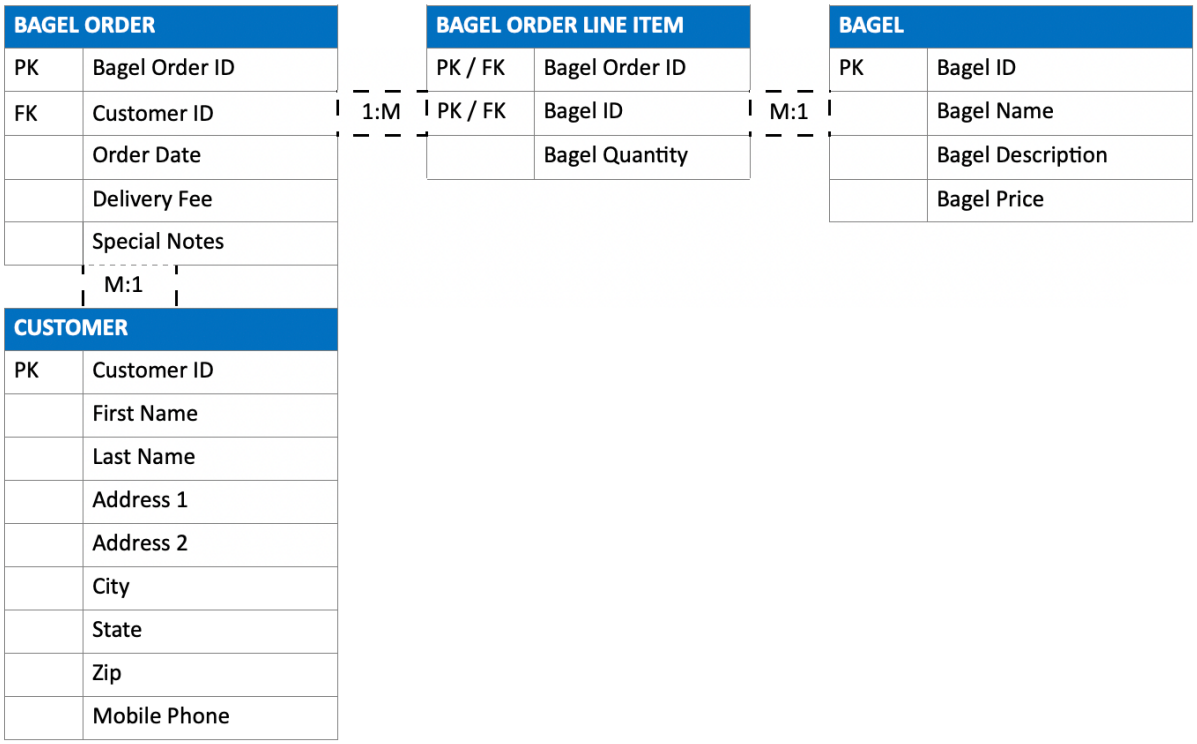
Cardinality of Relationships

Every single Bagel Order can have many Bagel Order Line Items but each Bagel Order Line Item can only be assigned to one Bagel Order. The Bagel Order Table has a one-to-many relationship with the Bagel Order Line Item table.

Every single Bagel Order Line Item must have one Bagel but any single Bagel can be in many Bagel Order Line Items. The Bagel order Line Item table has a many-to-one relationship with the Bagel table.

A2: Third Normal Form

Third Normal Form (3NF)



Attribute Explanation

The Bagel Order table features the Bagel Order Id, Customer ID, Order Date, Delivery Fee, and Special Notes attributes since these attributes are all dependent on the Bagel Order ID.

The Customer table features the Customer ID, First Name, Last Name, Address 1, Address 2, City, State, Zip, and Mobile Phone attributes since these attributes are all dependent on and can be referenced by just the Customer ID without needing a Bagel Order ID.

The Bagel Order Line Item table features the Bagel Order ID, Bagel ID, and Bagel Quantity since the Bagel Quantity is dependent on the unique compound attribute consisting of the Bagel Order ID and the Bagel ID.

The Bagel table features the Bagel ID, Bagel Name, Bagel Description, and Bagel Price attributes since these attributes are dependent on and can be referenced using just the Bagel ID.

Cardinality of Relationships

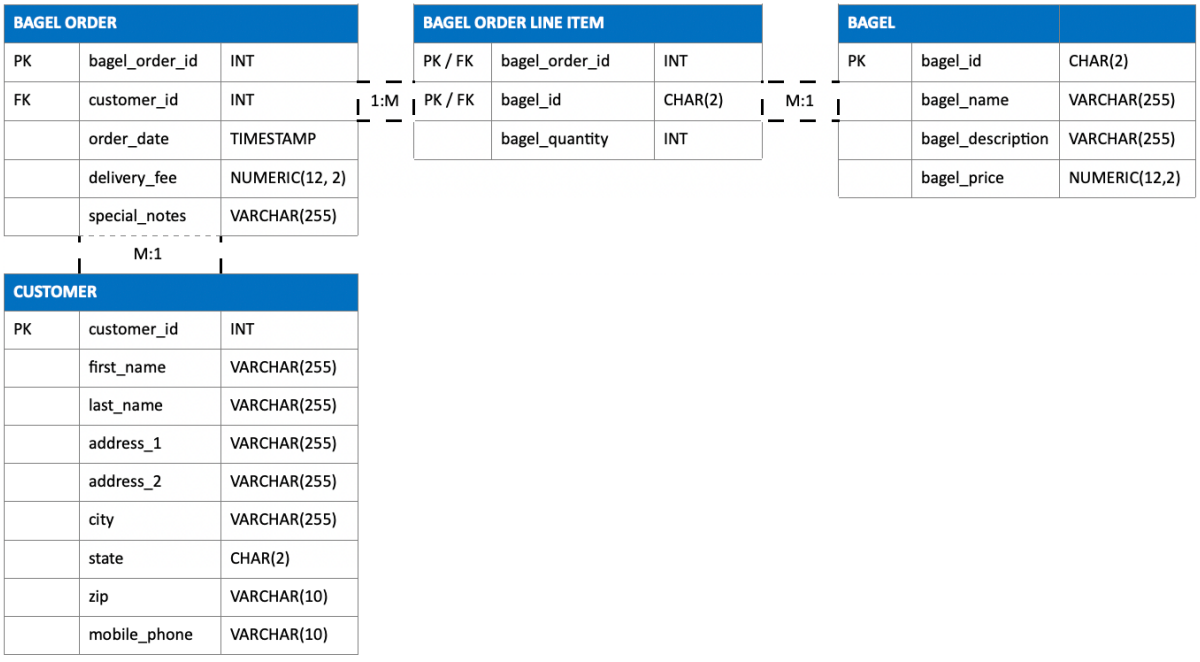
Every single Bagel Order can have many Bagel Order Line Items but each Bagel Order Line Item can only be assigned to one Bagel Order. The Bagel Order Table has a one-to-many relationship with the Bagel Order Line Item table.

Every single Bagel Order Line Item must have one Bagel but any single Bagel can be in many Bagel Order Line Items. The Bagel order Line Item table has a many-to-one relationship with the Bagel table.

Every single Bagel Order must have one Customer but any single Customer can have many Bagel Orders. The Bagel Order Table has a many-to-one relationship with the Customer table.

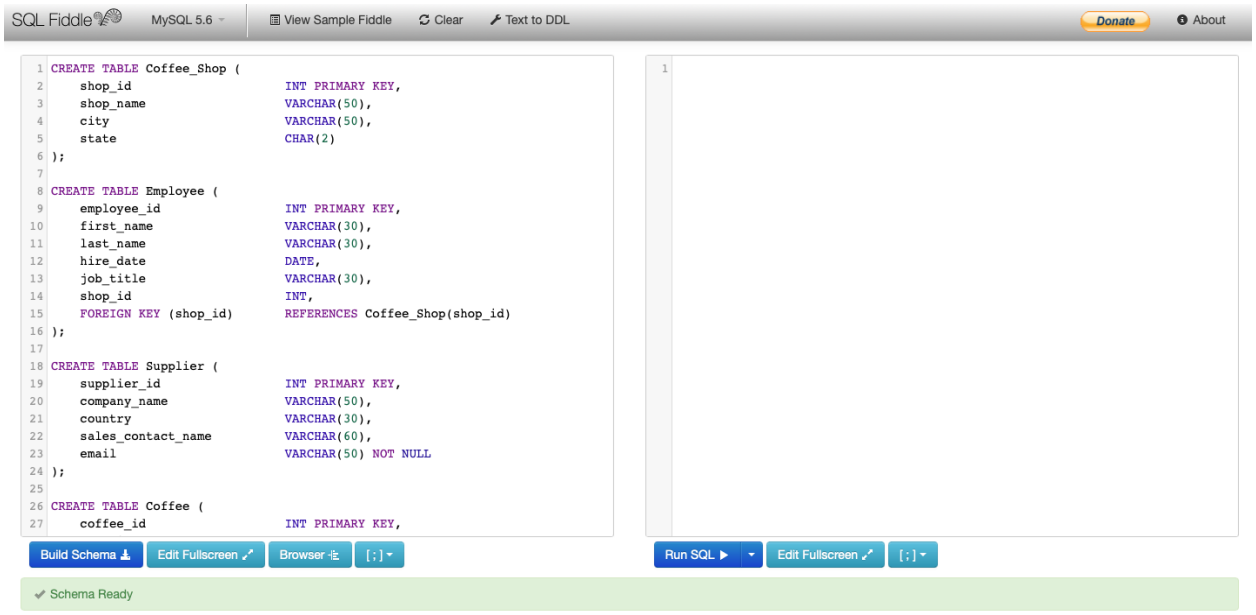
A3: Final Physical Database Model

Final Physical Database Model



Part B:Jaunty Coffee Co

B1:




```
CREATE TABLE Coffee_Shop (
    shop_id                     INT PRIMARY KEY,
    shop_name                   VARCHAR(50),
    city                        VARCHAR(50),
    state                       CHAR(2)
);

CREATE TABLE Employee (
    employee_id                 INT PRIMARY KEY,
    first_name                  VARCHAR(30),
    last_name                   VARCHAR(30),
    hire_date                   DATE,
    job_title                   VARCHAR(30),
    shop_id                     INT,
    FOREIGN KEY (shop_id)       REFERENCES Coffee_Shop(shop_id)
);

CREATE TABLE Supplier (
    supplier_id                 INT PRIMARY KEY,
    company_name                VARCHAR(50),
    country                     VARCHAR(30),
```

```
   sales_contact_name          VARCHAR(60),
   email                       VARCHAR(50) NOT NULL
);

CREATE TABLE Coffee (
   coffee_id                   INT PRIMARY KEY,
   shop_id                     INT,
   supplier_id                 INT,
   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:

SQL Fiddle | MySQL 5.6 | View Sample Fiddle | Clear | Text to DDL | Donate | About

```
    FOREIGN KEY (supplier_id)   REFERENCES Supplier(supplier_id)
);


INSERT INTO Coffee_Shop
VALUES
    (1, 'Starbucks Coffee', 'San Jose', 'CA'),
    (2, 'Pete Coffee', 'Los Angeles', 'CA'),
    (3, 'Phil Coffee',  'Brooklyn', 'NY');

INSERT INTO Employee
VALUES
    (1, 'John', 'Wick', '2023-09-10', 'Barista', 1),
    (2, 'Indiana', 'Jones', '2023-10-09', 'Manager', 2),
    (3, 'Sherlock', 'Holmes', '2023-07-12', 'Driver', 3);

INSERT INTO Supplier
VALUES
    (1, 'CoffeeBeans', 'United States', 'Michael West', 'michael@coffeebeans.
    (2, 'ValueBeans', 'United States', 'John Simmons', 'john@valuebeans.com')
    (3, 'LuxBeans', 'United States', 'Mary Strong', 'mary@luxbeans.com');

INSERT INTO Coffee
VALUES
    (1, 1, 1, 'Mocha', 7.99),
    (2, 2, 2, 'Frap', 5.99),
    (3, 3, 3, 'Black', 1.50);
```

Build Schema | Edit Fullscreen | Browser | [;]

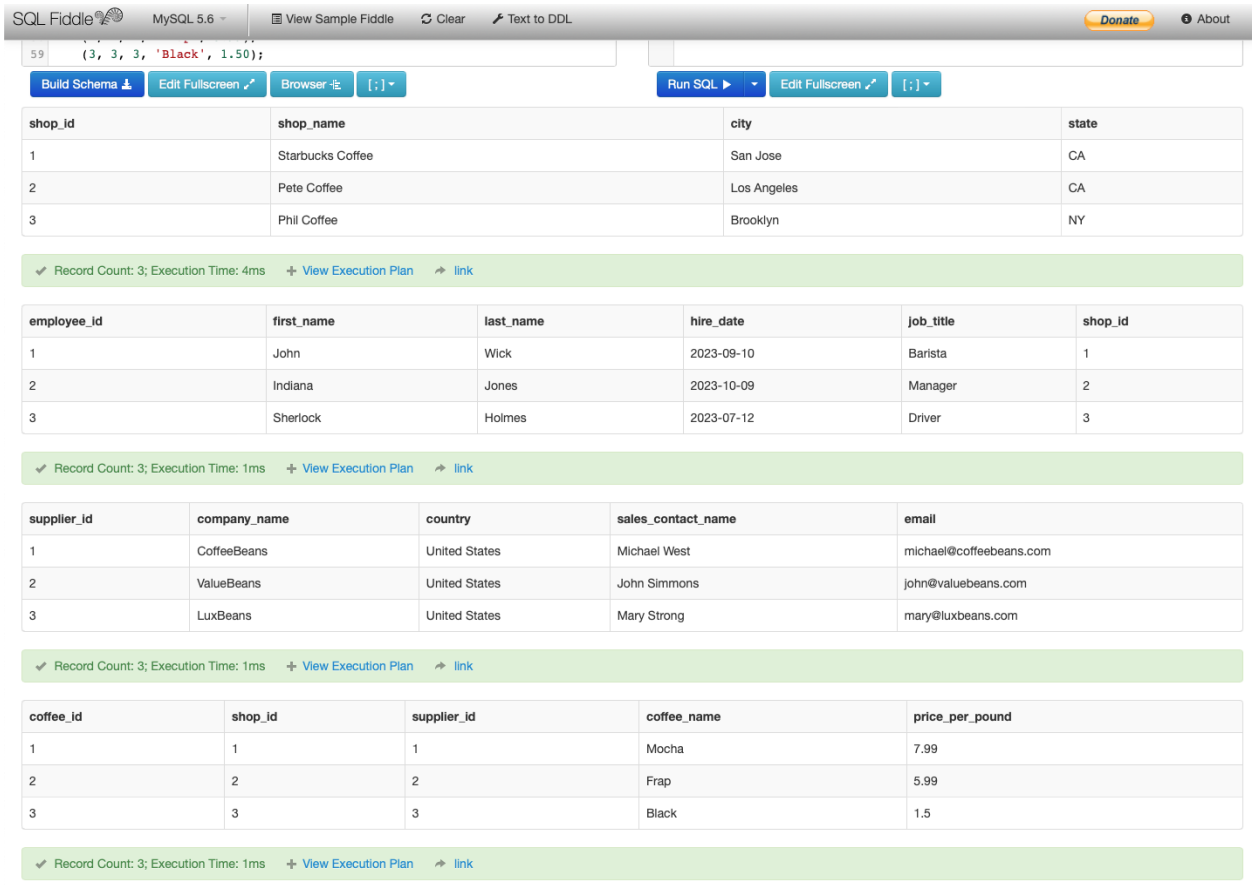
```
SELECT * FROM Coffee_shop;
SELECT * FROM Employee;
SELECT * FROM Supplier;
SELECT * FROM Coffee;
```

Run SQL | Edit Fullscreen | [;]

shop_id	shop_name	city	state
1	Starbucks Coffee	San Jose	CA
2	Pete Coffee	Los Angeles	CA
3	Phil Coffee	Brooklyn	NY

Record Count: 3; Execution Time: 4ms | View Execution Plan | link

employee_id	first_name	last_name	hire_date	job_title	shop_id
1	John	Wick	2023-09-10	Barista	1
2	Indiana	Jones	2023-10-09	Manager	2
3	Sherlock	Holmes	2023-07-12	Driver	3



shop_id	shop_name	city	state
1	Starbucks Coffee	San Jose	CA
2	Pete Coffee	Los Angeles	CA
3	Phil Coffee	Brooklyn	NY

Record Count: 3; Execution Time: 4ms View Execution Plan link

employee_id	first_name	last_name	hire_date	job_title	shop_id
1	John	Wick	2023-09-10	Barista	1
2	Indiana	Jones	2023-10-09	Manager	2
3	Sherlock	Holmes	2023-07-12	Driver	3

Record Count: 3; Execution Time: 1ms View Execution Plan link

supplier_id	company_name	country	sales_contact_name	email
1	CoffeeBeans	United States	Michael West	michael@coffeebeans.com
2	ValueBeans	United States	John Simmons	john@valuebeans.com
3	LuxBeans	United States	Mary Strong	mary@luxbeans.com

Record Count: 3; Execution Time: 1ms View Execution Plan link

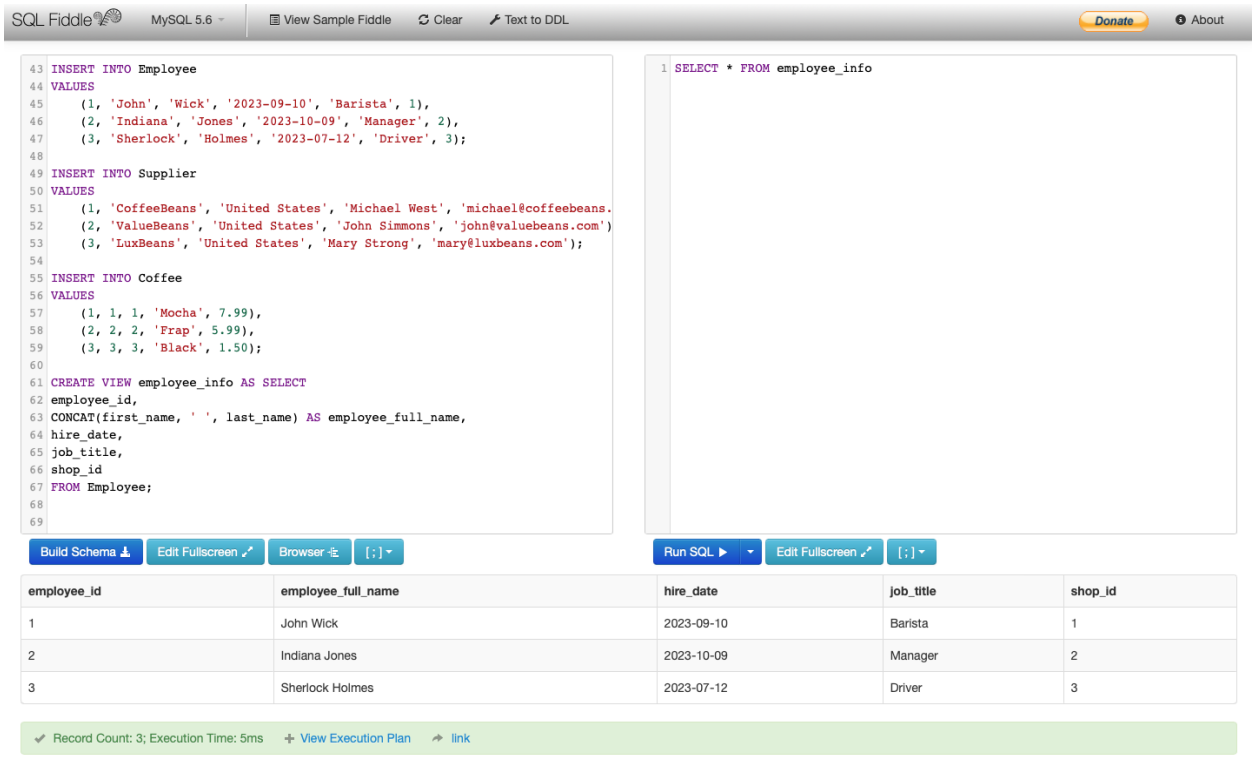
coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
1	1	1	Mocha	7.99
2	2	2	Frap	5.99
3	3	3	Black	1.5

Record Count: 3; Execution Time: 1ms View Execution Plan link

```sql
INSERT INTO Coffee_Shop
VALUES
   (1, 'Starbucks Coffee', 'San Jose', 'CA'),
   (2, 'Pete Coffee', 'Los Angeles', 'CA'),
   (3, 'Phil Coffee', 'Brooklyn', 'NY');

INSERT INTO Employee
VALUES
   (1, 'John', 'Wick', '2023-09-10', 'Barista', 1),
   (2, 'Indiana', 'Jones', '2023-10-09', 'Manager', 2),
   (3, 'Sherlock', 'Holmes', '2023-07-12', 'Driver', 3);

INSERT INTO Supplier
VALUES
   (1, 'CoffeeBeans', 'United States', 'Michael West', 'michael@coffeebeans.com'),
   (2, 'ValueBeans', 'United States', 'John Simmons', 'john@valuebeans.com'),
   (3, 'LuxBeans', 'United States', 'Mary Strong', 'mary@luxbeans.com');

INSERT INTO Coffee
VALUES
```

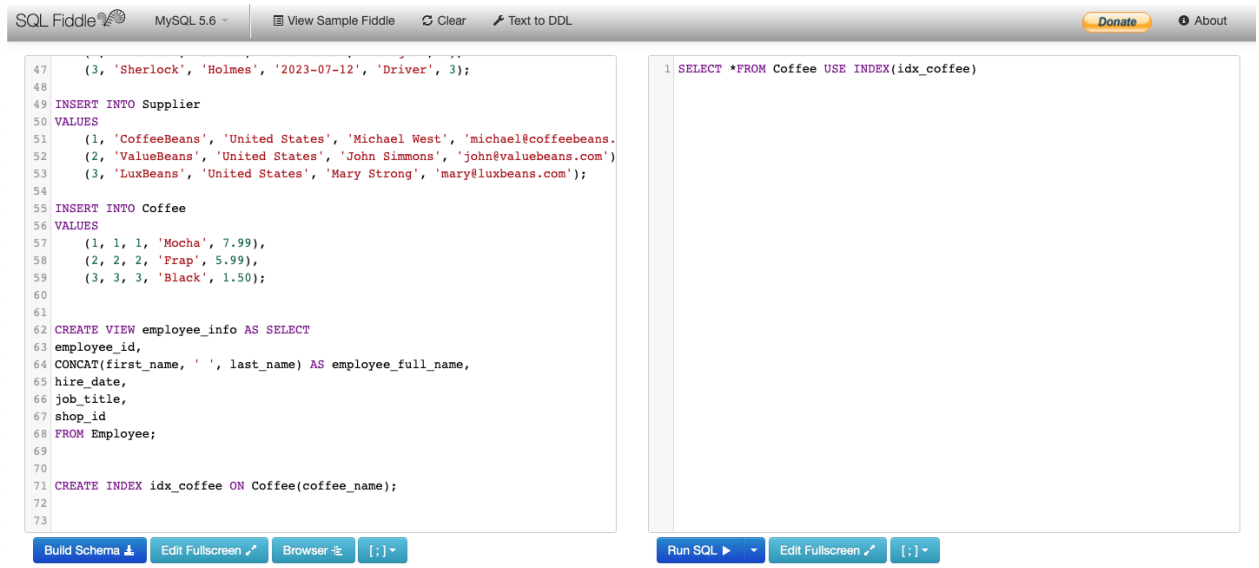
```
    (1, 1, 1, 'Mocha', 7.99),
    (2, 2, 2, 'Frap', 5.99),
    (3, 3, 3, 'Black', 1.50);
```

B3:



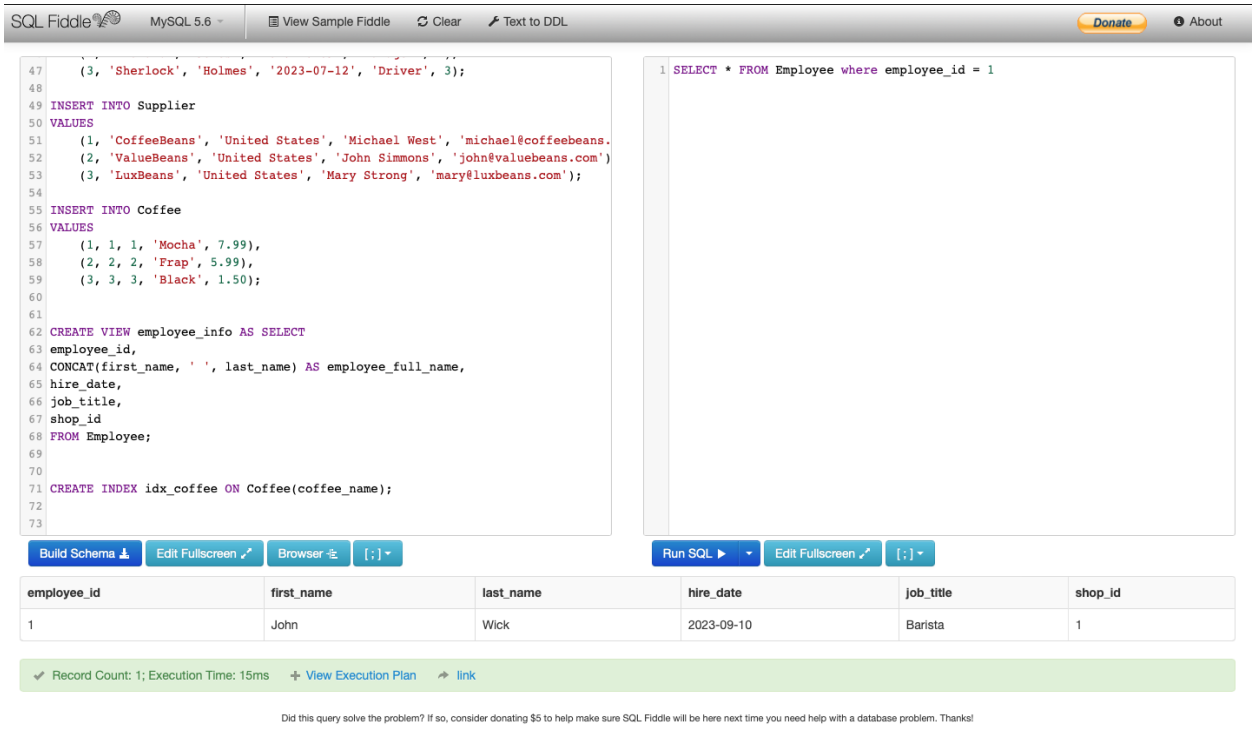

```
CREATE VIEW employee_info AS SELECT
employee_id,
CONCAT(first_name, ' ', last_name) AS employee_full_name,
hire_date,
job_title,
shop_id
FROM Employee;
```

B4:



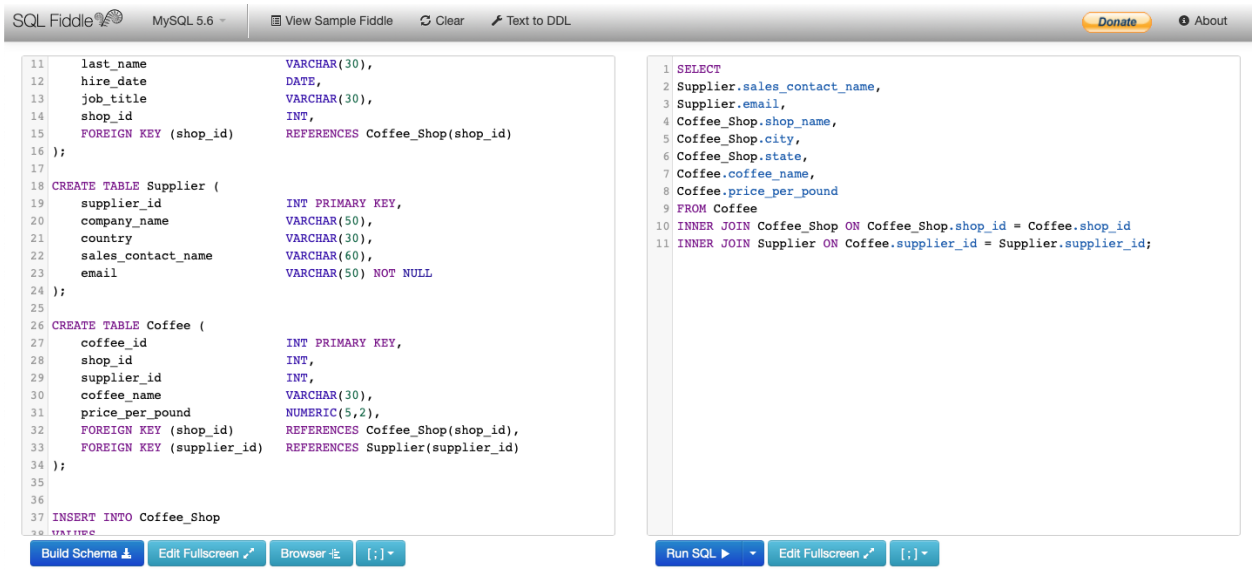

```
CREATE INDEX idx_coffee ON Coffee(coffee_name);
```

B5:




```
SELECT * FROM Employee where employee_id = 1
```

B6:




```sql
SELECT
Supplier.sales_contact_name,
Supplier.email,
Coffee_Shop.shop_name,
Coffee_Shop.city,
Coffee_Shop.state,
Coffee.coffee_name,
Coffee.price_per_pound
FROM Coffee
INNER JOIN Coffee_Shop ON Coffee_Shop.shop_id = Coffee.shop_id
INNER JOIN Supplier ON Coffee.supplier_id = Supplier.supplier_id;
```