Jarone McCorkle
C170: Data Management – Applications (MySQL)

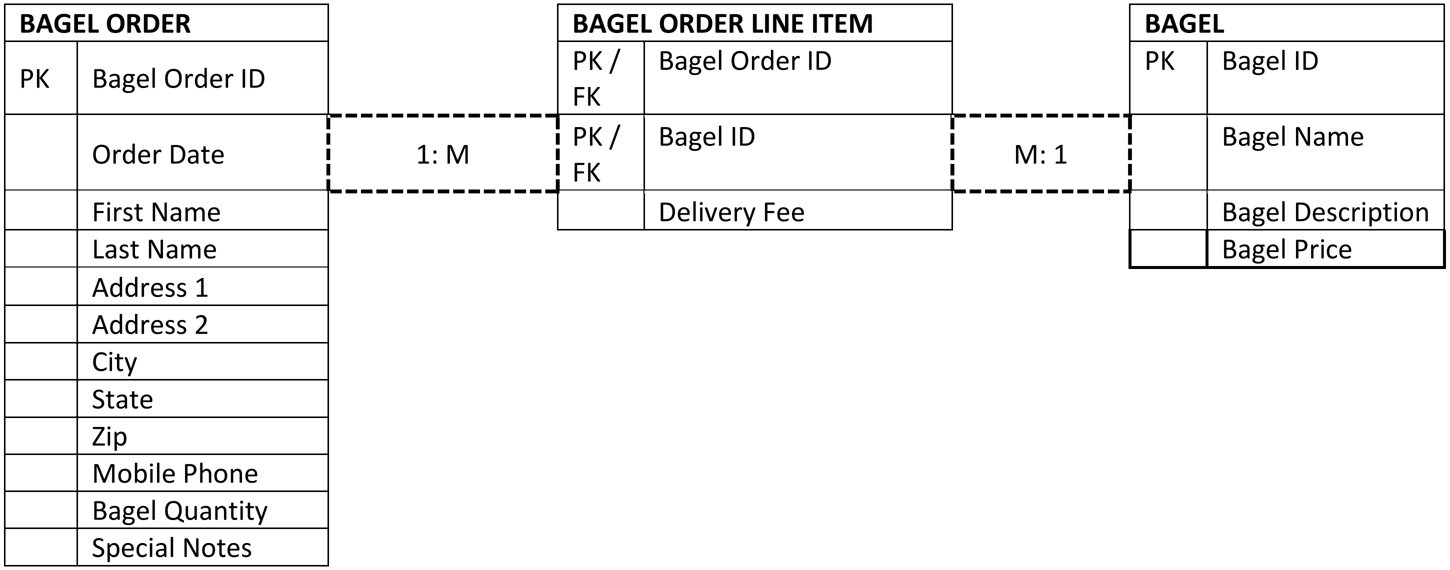
NORMALIZATION AND DATABASE DESIGN

A. Database Model representing Nora’s Bagel Bin:

1.

a. **Second Normal Form (2NF)**

b.



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

BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Delivery Fee

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

(cont’d)

b. **Description (Cardinality):**

Bagel orders can have only one and only one bagel order line items, while bagel order line items may have many bagel orders. Similarly, bagels can have only one and only one bagel order line items, while a bagel order line item can have many bagels.

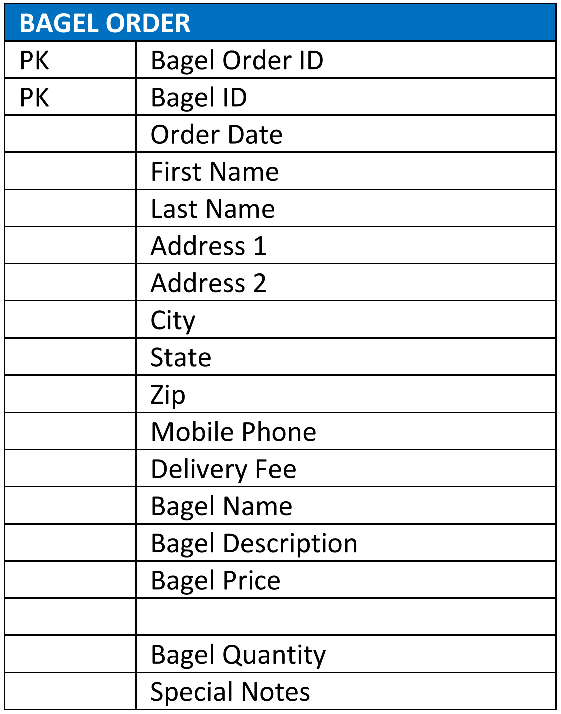
c. **Explanation of a and b:**

In order to normalize the database to second normal form, it would already have to be in first normal form. In addition to this, for the database to be in second normal form, we would also need to remove all partial dependencies. Which means that non-key attributes should be fully dependent on the candidate key.

(cont’d)

Nora’s Bagel Bin Database Blueprints

First Normal Form (1NF)



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

Figure 1

This will make more sense if we reference figure 1, which is our database in first normal form. Looking at figure one, we have non-key attributes that are not fully dependent on both candidate keys: Bagel Order ID, and Bagel ID. These attributes need to be separated to their own tables so that all non-key attributes are fully dependent on the candidate key within their respective tables.

Therefore, figure one, the Bagel Order table, can be broken down into two additional tables; Bagel Order Line Form, and Bagel. Now the tables are each narrowed down to a single purpose; For example, the Bagel table has all attributes related to, you guessed it, a bagel. With the bagel name, description, price per bagel, and a bagel ID set as the primary key to differentiate each bagel from one another. We can clearly see that attributes not related to a bagel, would not be in this table. This is what we mean by single purpose.

Lastly, the relationships between the entities are such that for every one bagel order, there can be many line items, and each line item has a description of a bagel.

(cont'd)

2. **Database Normalized to third normal form**

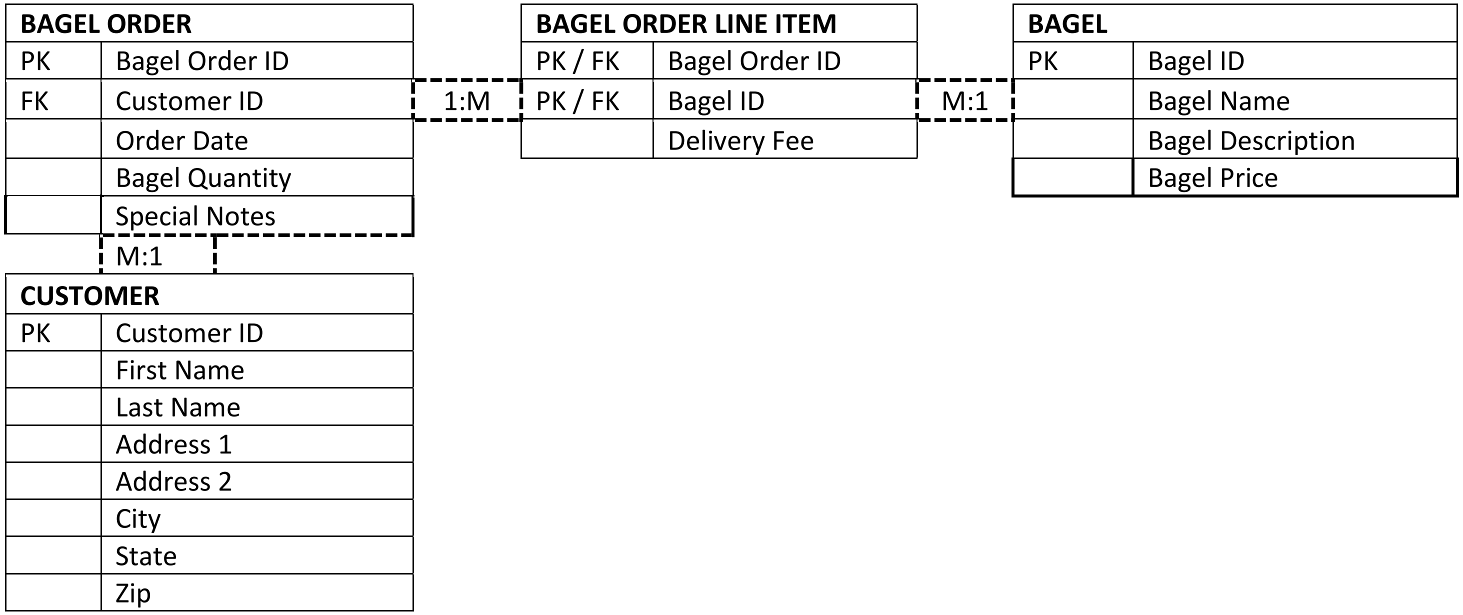
Nora's Bagel Bin Database Blueprints

Third Normal Form (3NF)

a. **Attribute assignment b. Table names and c. Foreign keys used to link each table:**

d. **Description (Cardinality):**

Each customer can have many bagel orders. While each bagel order can have only one and only one bagel order line items. Bagel order line items may have many bagel orders. Similarly, bagels can have only one and only one bagel order line items, while a bagel order line item can have many bagels.



BAGEL ORDER	
PK	Bagel Order ID
FK	Customer ID
	Order Date
	Bagel Quantity
	Special Notes

BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Delivery Fee

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

CUSTOMER	
PK	Customer ID
	First Name
	Last Name
	Address 1
	Address 2
	City
	State
	Zip

e. **Explanation of a and b:**

The database was normalized to third normal form by meeting all of the requirements of second normal form. Secondly, all transitive dependencies were removed from tables; which means looking at each table to see if more fields that aren't dependent on a key can be split into other tables. Hence, a customer table was created with its own private key, and added a foreign key to the bagel order to connect the two tables.

In terms of cardinality from customer to bagel:

Each customer can have many bagel orders. While each bagel order can have only one and only one bagel order line items. Bagel order line items may have many bagel orders. Similarly, bagels can have only one and only one bagel order line items, while a bagel order line item can have many bagels.

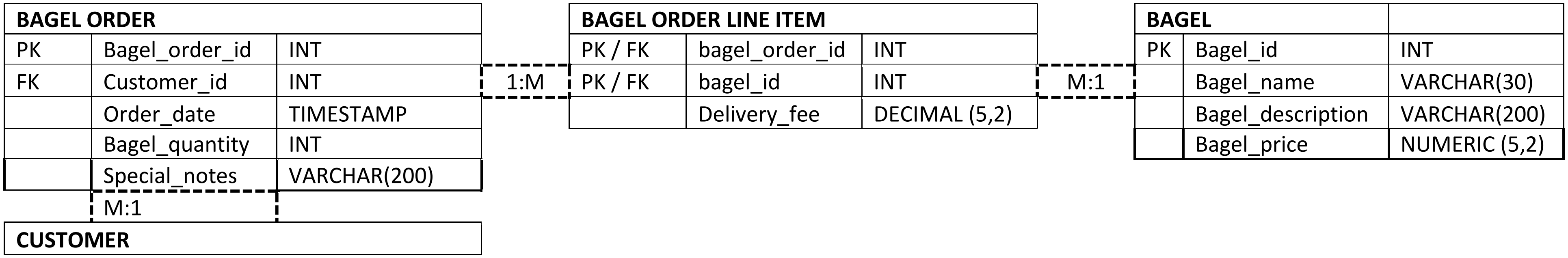
(cont'd)

3. **Final Database Model:**

Nora's Bagel Bin Database Blueprints

Final Physical Database Model

a. **Names and cardinality transferred from 2NF table to 3NF table**
b. **b. Assign datatypes to each table**



BAGEL ORDER		
PK	Bagel_order_id	INT
FK	Customer_id	INT
	Order_date	TIMESTAMP
	Bagel_quantity	INT
	Special_notes	VARCHAR(200)

1:M

BAGEL ORDER LINE ITEM		
PK / FK	bagel_order_id	INT
PK / FK	bagel_id	INT
	Delivery_fee	DECIMAL (5,2)

M:1

BAGEL		
PK	Bagel_id	INT
	Bagel_name	VARCHAR(30)
	Bagel_description	VARCHAR(200)
	Bagel_price	NUMERIC (5,2)

M:1

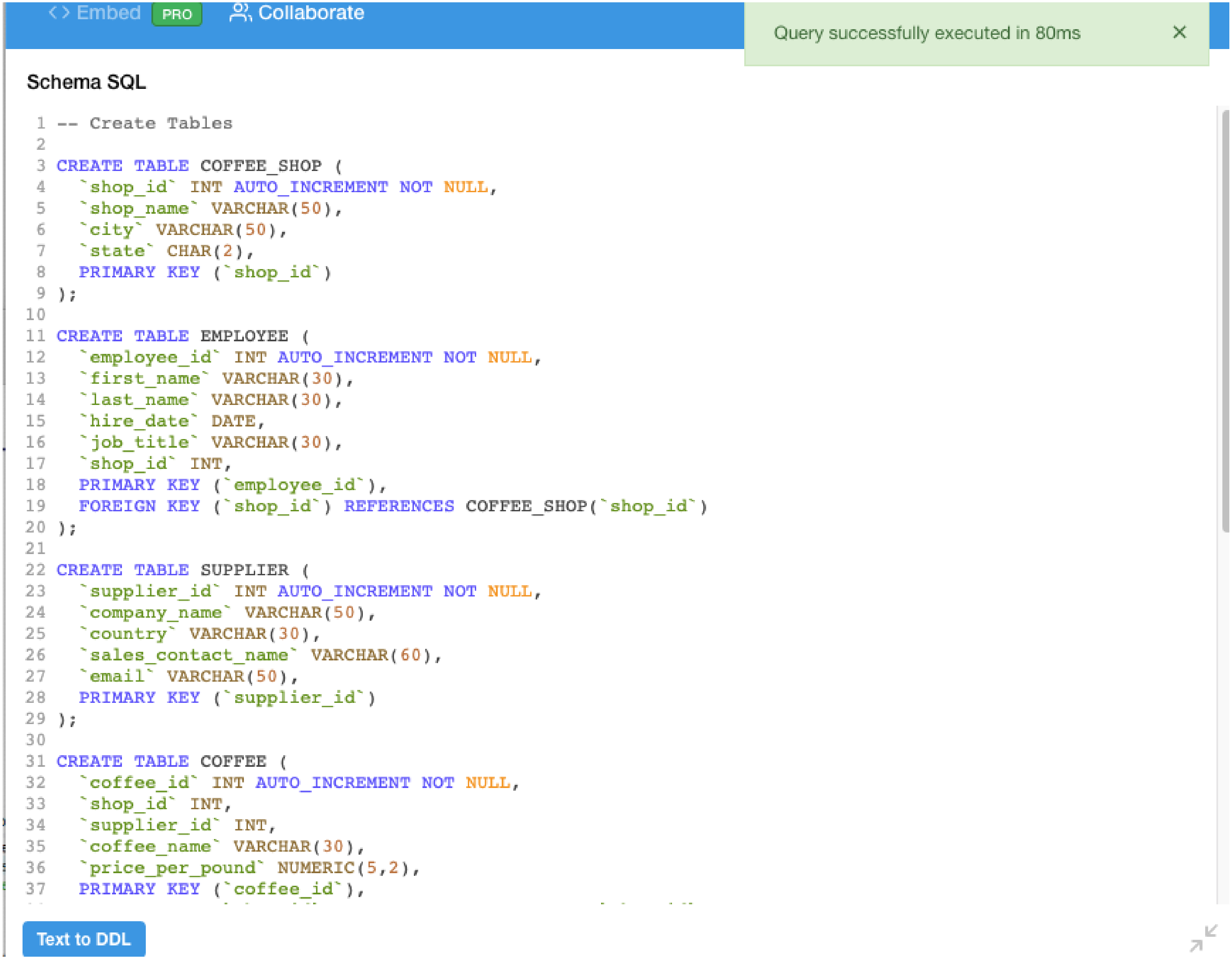
CUSTOMER

PK	Customer_id	INT
	First_name	VARCHAR(30)
	Last_name	VARCHAR(30)
	Address 1	VARCHAR(95)
	Address 2	VARCHAR(95)
	City	VARCHAR(35)
	State	CHAR(2)
	Zip	VARCHAR(5)
	Mobile_phone	VARCHAR(20)

B. Create Jaunty Coffee Co. Database

Proof of Execution Screenshots

1. Develop SQL code to create *each* table as specified in the attached "Jaunty Coffee Co. ERD"

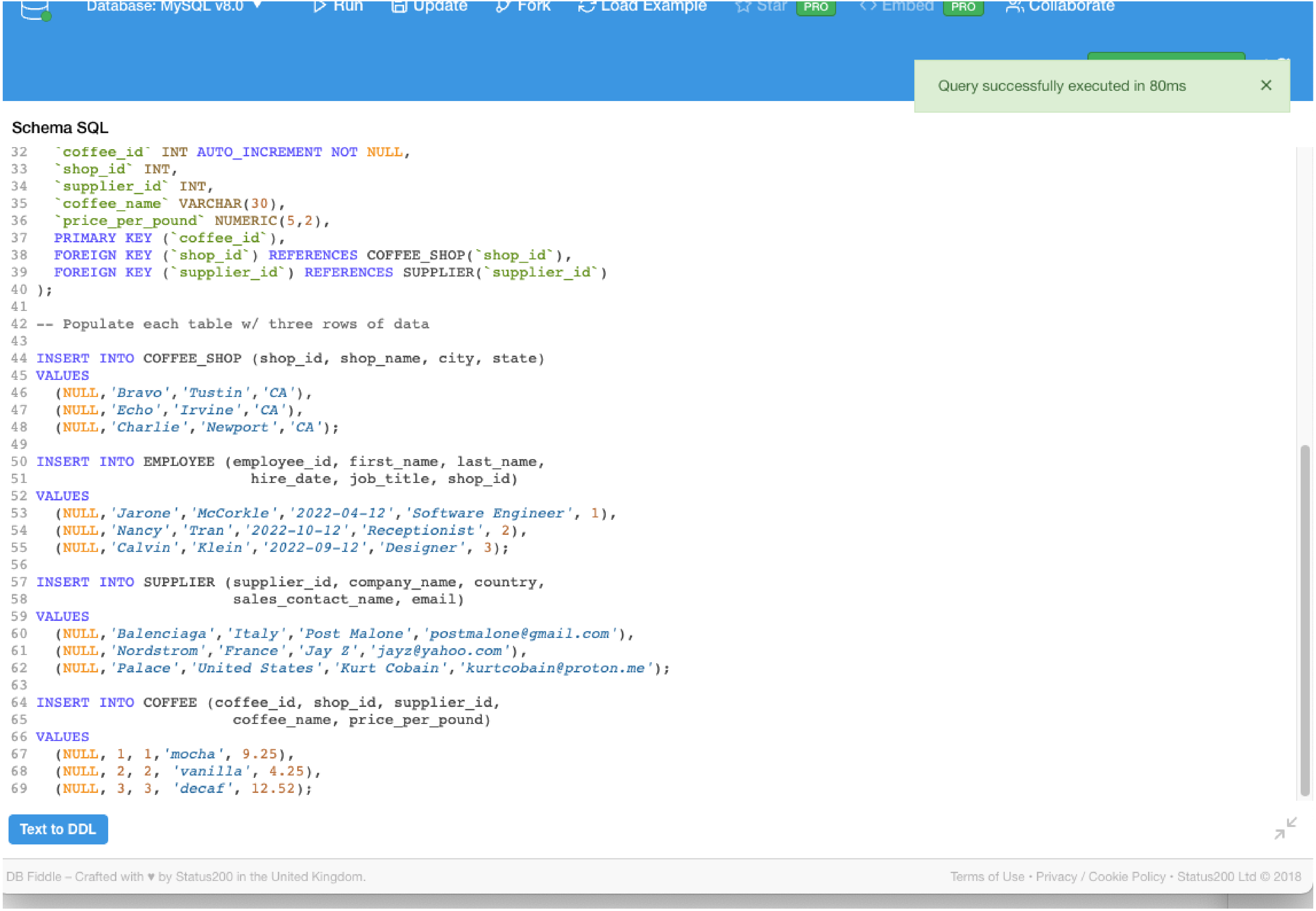


Query successfully executed in 80ms

Schema SQL

```
-- Create Tables

CREATE TABLE COFFEE_SHOP (
  `shop_id` INT AUTO_INCREMENT NOT NULL,
  `shop_name` VARCHAR(50),
  `city` VARCHAR(50),
  `state` CHAR(2),
  PRIMARY KEY (`shop_id`)
);

CREATE TABLE EMPLOYEE (
  `employee_id` INT AUTO_INCREMENT NOT NULL,
  `first_name` VARCHAR(30),
  `last_name` VARCHAR(30),
  `hire_date` DATE,
  `job_title` VARCHAR(30),
  `shop_id` INT,
  PRIMARY KEY (`employee_id`),
  FOREIGN KEY (`shop_id`) REFERENCES COFFEE_SHOP(`shop_id`)
);

CREATE TABLE SUPPLIER (
  `supplier_id` INT AUTO_INCREMENT NOT NULL,
  `company_name` VARCHAR(50),
  `country` VARCHAR(30),
  `sales_contact_name` VARCHAR(60),
  `email` VARCHAR(50),
  PRIMARY KEY (`supplier_id`)
);

CREATE TABLE COFFEE (
  `coffee_id` INT AUTO_INCREMENT NOT NULL,
  `shop_id` INT,
  `supplier_id` INT,
  `coffee_name` VARCHAR(30),
  `price_per_pound` NUMERIC(5,2),
  PRIMARY KEY (`coffee_id`),
```

2. Develop SQL code to populate at least **three** rows of data in each table.



Query successfully executed in 80ms

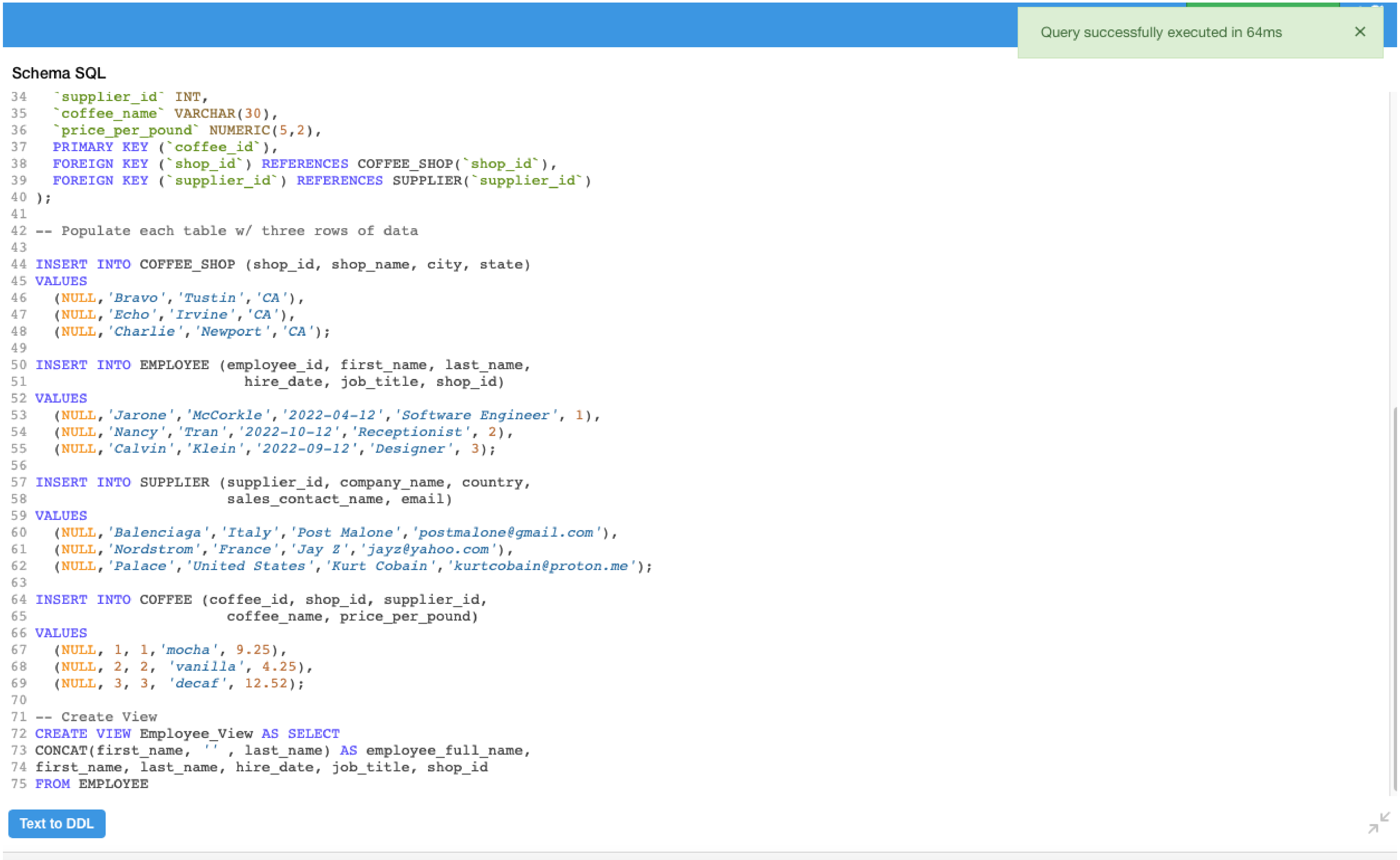
Schema SQL

```
  `coffee_id` INT AUTO_INCREMENT NOT NULL,
  `shop_id` INT,
  `supplier_id` INT,
  `coffee_name` VARCHAR(30),
  `price_per_pound` NUMERIC(5,2),
  PRIMARY KEY (`coffee_id`),
  FOREIGN KEY (`shop_id`) REFERENCES COFFEE_SHOP(`shop_id`),
  FOREIGN KEY (`supplier_id`) REFERENCES SUPPLIER(`supplier_id`)
);

-- Populate each table w/ three rows of data

INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state)
VALUES
  (NULL,'Bravo','Tustin','CA'),
  (NULL,'Echo','Irvine','CA'),
  (NULL,'Charlie','Newport','CA');

INSERT INTO EMPLOYEE (employee_id, first_name, last_name,
                        hire_date, job_title, shop_id)
VALUES
  (NULL,'Jarone','McCorkle','2022-04-12','Software Engineer', 1),
  (NULL,'Nancy','Tran','2022-10-12','Receptionist', 2),
  (NULL,'Calvin','Klein','2022-09-12','Designer', 3);

INSERT INTO SUPPLIER (supplier_id, company_name, country,
                      sales_contact_name, email)
VALUES
  (NULL,'Balenciaga','Italy','Post Malone','postmalone@gmail.com'),
  (NULL,'Nordstrom','France','Jay Z','jayz@yahoo.com'),
  (NULL,'Palace','United States','Kurt Cobain','kurtcobain@proton.me');

INSERT INTO COFFEE (coffee_id, shop_id, supplier_id,
                      coffee_name, price_per_pound)
VALUES
  (NULL, 1, 1,'mocha', 9.25),
  (NULL, 2, 2, 'vanilla', 4.25),
  (NULL, 3, 3, 'decaf', 12.52);
```

DB Fiddle – Crafted with ♥ by Status200 in the United Kingdom.

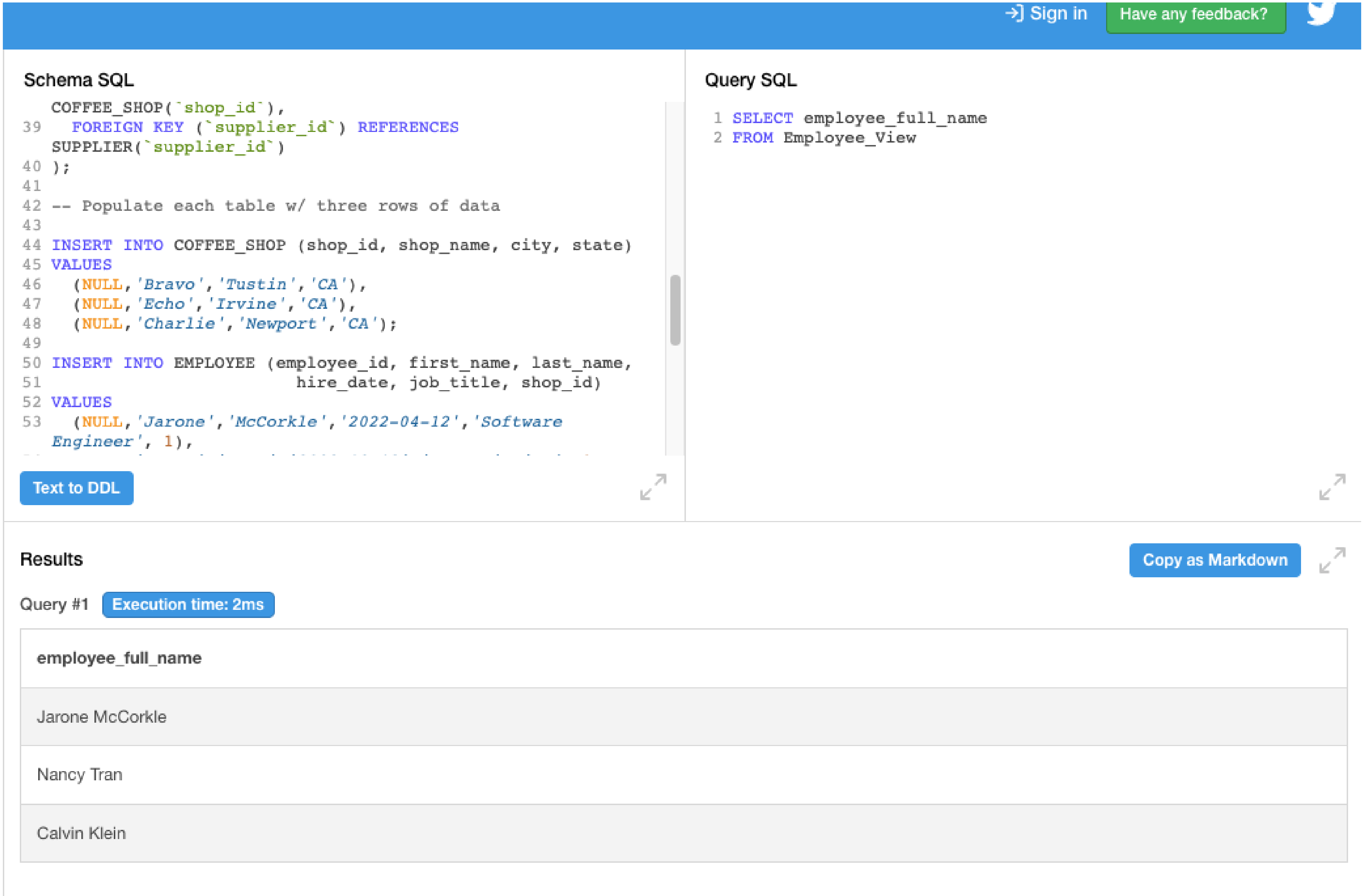
Terms of Use • Privacy / Cookie Policy • Status200 Ltd © 2018

3. **Develop SQL code to create a view, concatenate employees first and last name**



Query successfully executed in 64ms

```sql
  `supplier_id` INT,
  `coffee_name` VARCHAR(30),
  `price_per_pound` NUMERIC(5,2),
  PRIMARY KEY (`coffee_id`),
  FOREIGN KEY (`shop_id`) REFERENCES COFFEE_SHOP(`shop_id`),
  FOREIGN KEY (`supplier_id`) REFERENCES SUPPLIER(`supplier_id`)
);

-- Populate each table w/ three rows of data

INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state)
VALUES
  (NULL,'Bravo','Tustin','CA'),
  (NULL,'Echo','Irvine','CA'),
  (NULL,'Charlie','Newport','CA');

INSERT INTO EMPLOYEE (employee_id, first_name, last_name,
                        hire_date, job_title, shop_id)
VALUES
  (NULL,'Jarone','McCorkle','2022-04-12','Software Engineer', 1),
  (NULL,'Nancy','Tran','2022-10-12','Receptionist', 2),
  (NULL,'Calvin','Klein','2022-09-12','Designer', 3);

INSERT INTO SUPPLIER (supplier_id, company_name, country,
                      sales_contact_name, email)
VALUES
  (NULL,'Balenciaga','Italy','Post Malone','postmalone@gmail.com'),
  (NULL,'Nordstrom','France','Jay Z','jayz@yahoo.com'),
  (NULL,'Palace','United States','Kurt Cobain','kurtcobain@proton.me');

INSERT INTO COFFEE (coffee_id, shop_id, supplier_id,
                      coffee_name, price_per_pound)
VALUES
  (NULL, 1, 1,'mocha', 9.25),
  (NULL, 2, 2, 'vanilla', 4.25),
  (NULL, 3, 3, 'decaf', 12.52);

-- Create View
CREATE VIEW Employee_View AS SELECT
CONCAT(first_name, ' ' , last_name) AS employee_full_name,
first_name, last_name, hire_date, job_title, shop_id
FROM EMPLOYEE
```



Schema SQL

```
COFFEE_SHOP(`shop_id`),
  FOREIGN KEY (`supplier_id`) REFERENCES
SUPPLIER(`supplier_id`)
);

-- Populate each table w/ three rows of data

INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state)
VALUES
  (NULL,'Bravo','Tustin','CA'),
  (NULL,'Echo','Irvine','CA'),
  (NULL,'Charlie','Newport','CA');

INSERT INTO EMPLOYEE (employee_id, first_name, last_name,
                      hire_date, job_title, shop_id)
VALUES
  (NULL,'Jarone','McCorkle','2022-04-12','Software
Engineer', 1),
```

Query SQL

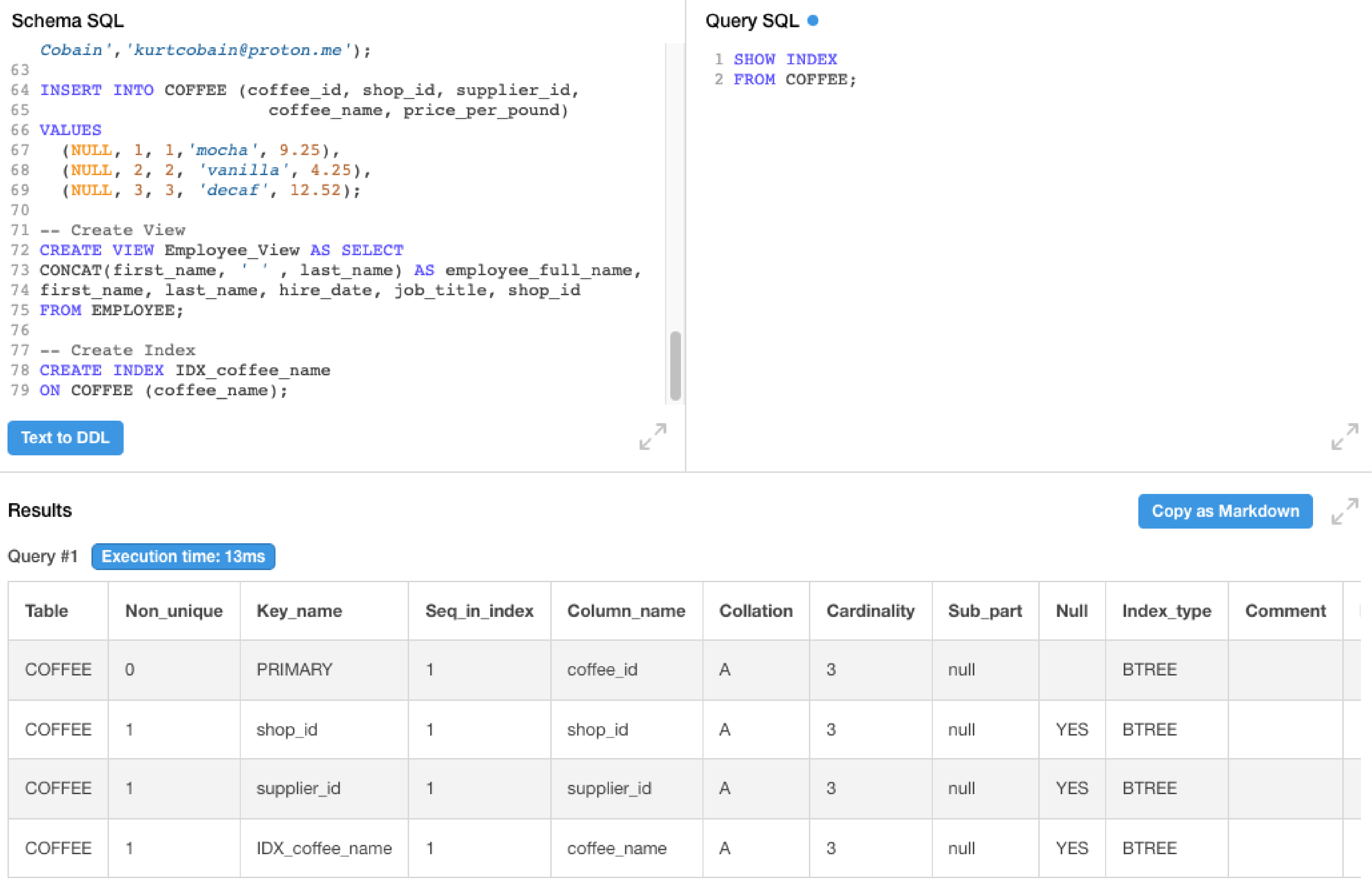
```
SELECT employee_full_name
FROM Employee_View
```

Results

Query #1 Execution time: 2ms

employee_full_name
Jarone McCorkle
Nancy Tran
Calvin Klein

4. **Develop SQL code to create an index on the coffee_name field from the Coffee table.**

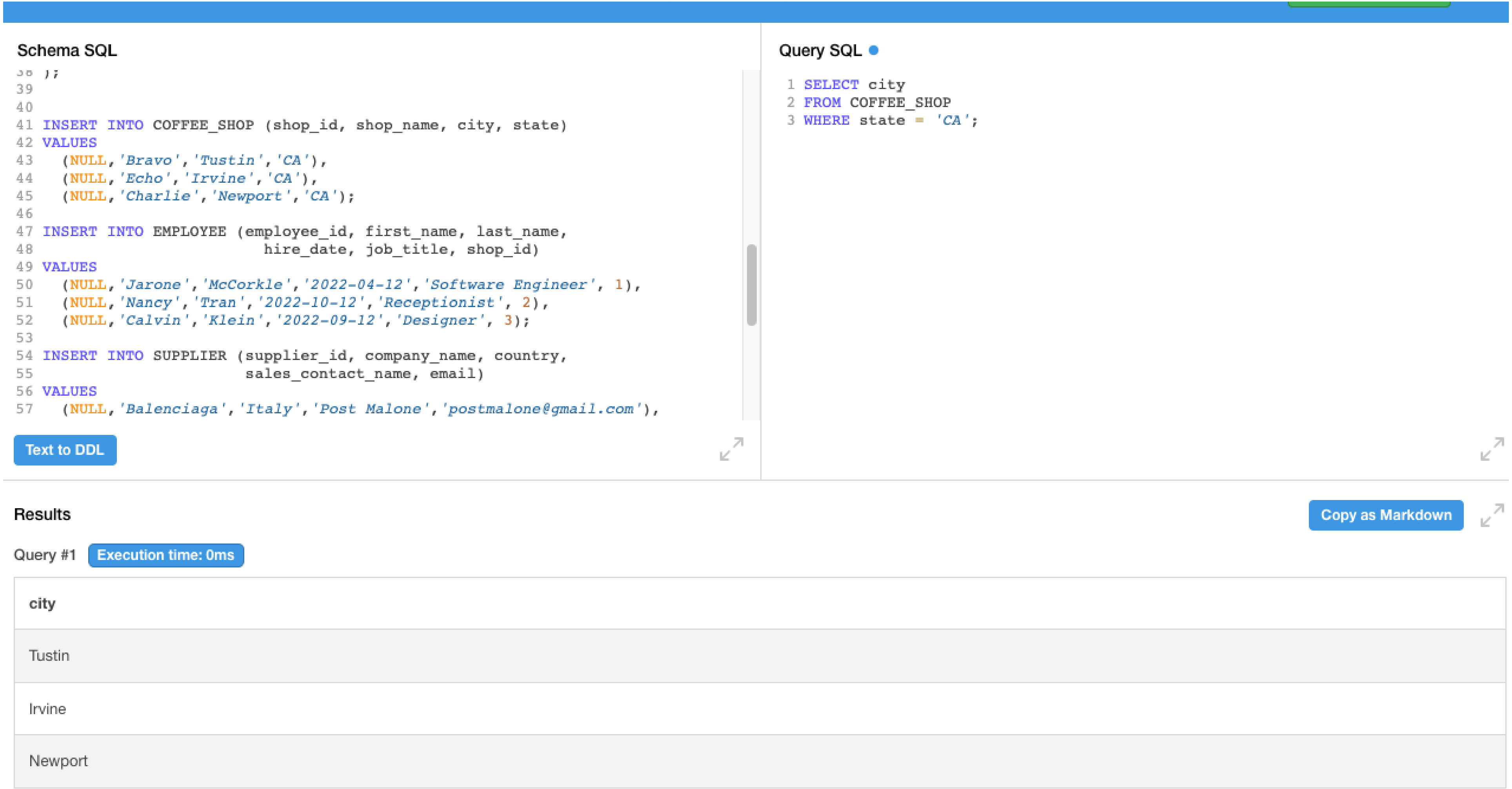


```sql
Cobain','kurtcobain@proton.me');

INSERT INTO COFFEE (coffee_id, shop_id, supplier_id,
                      coffee_name, price_per_pound)
VALUES
  (NULL, 1, 1,'mocha', 9.25),
  (NULL, 2, 2, 'vanilla', 4.25),
  (NULL, 3, 3, 'decaf', 12.52);

-- Create View
CREATE VIEW Employee_View AS SELECT
CONCAT(first_name, ' ' , last_name) AS employee_full_name,
first_name, last_name, hire_date, job_title, shop_id
FROM EMPLOYEE;

-- Create Index
CREATE INDEX IDX_coffee_name
ON COFFEE (coffee_name);
```

```sql
SHOW INDEX
FROM COFFEE;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Null	Index_type	Comment
COFFEE	0	PRIMARY	1	coffee_id	A	3	null		BTREE	
COFFEE	1	shop_id	1	shop_id	A	3	null	YES	BTREE	
COFFEE	1	supplier_id	1	supplier_id	A	3	null	YES	BTREE	
COFFEE	1	IDX_coffee_name	1	coffee_name	A	3	null	YES	BTREE	

5. Develop SQL code to create an SFW (SELECT–FROM–WHERE) query for *any* of your tables or views

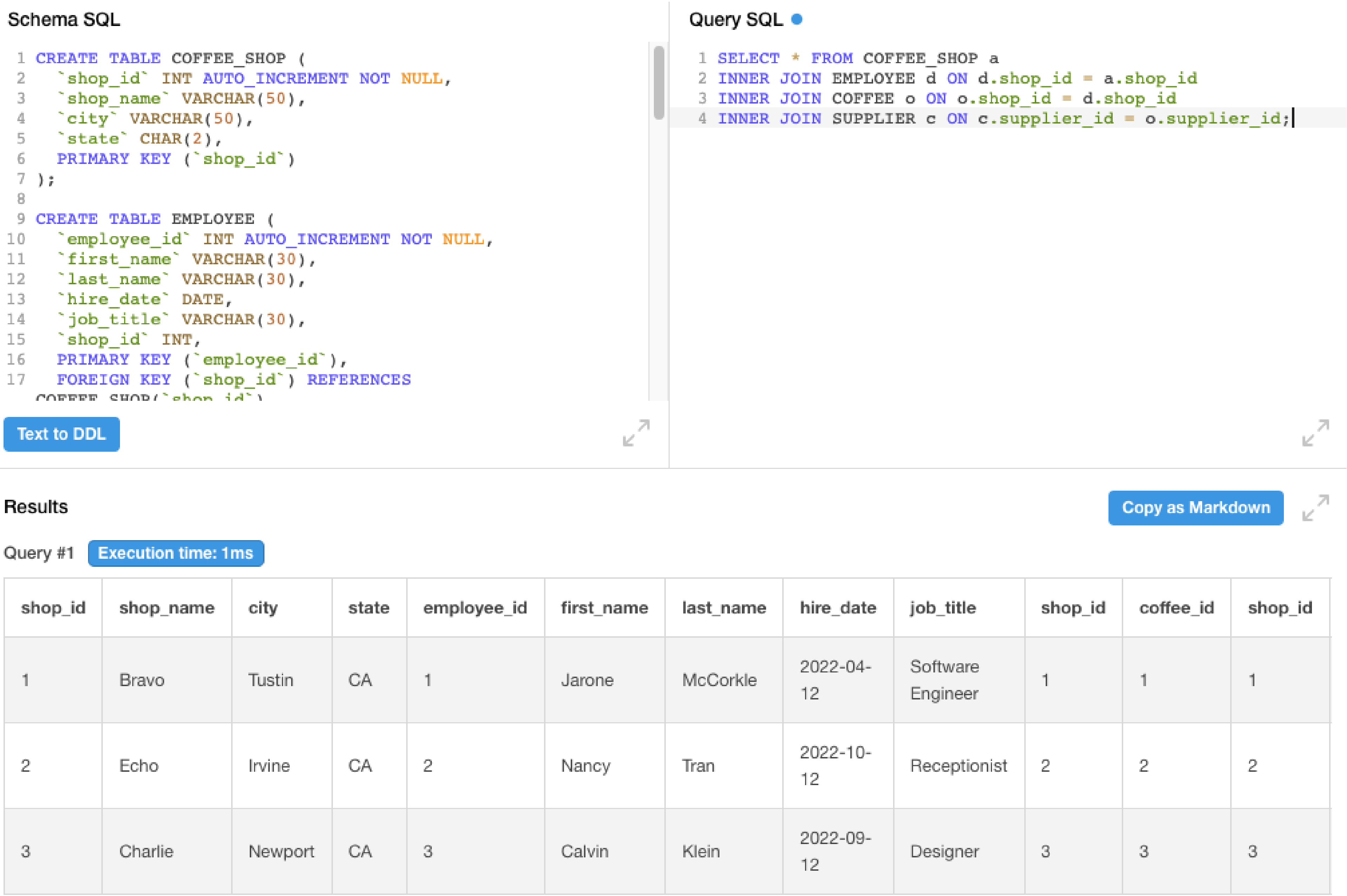


```
);


INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state)
VALUES
  (NULL,'Bravo','Tustin','CA'),
  (NULL,'Echo','Irvine','CA'),
  (NULL,'Charlie','Newport','CA');

INSERT INTO EMPLOYEE (employee_id, first_name, last_name,
                        hire_date, job_title, shop_id)
VALUES
  (NULL,'Jarone','McCorkle','2022-04-12','Software Engineer', 1),
  (NULL,'Nancy','Tran','2022-10-12','Receptionist', 2),
  (NULL,'Calvin','Klein','2022-09-12','Designer', 3);

INSERT INTO SUPPLIER (supplier_id, company_name, country,
                      sales_contact_name, email)
VALUES
  (NULL,'Balenciaga','Italy','Post Malone','postmalone@gmail.com'),
```

```
SELECT city
FROM COFFEE_SHOP
WHERE state = 'CA';
```

city
Tustin
Irvine
Newport

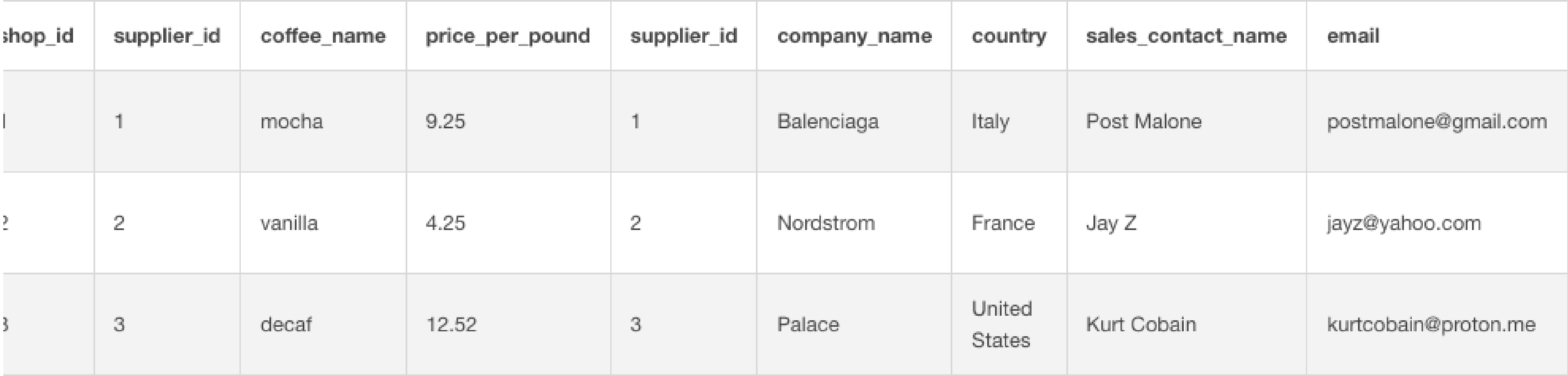
6. Join **three** different tables and include attributes from *all* three tables in its output



```sql
CREATE TABLE COFFEE_SHOP (
  `shop_id` INT AUTO_INCREMENT NOT NULL,
  `shop_name` VARCHAR(50),
  `city` VARCHAR(50),
  `state` CHAR(2),
  PRIMARY KEY (`shop_id`)
);

CREATE TABLE EMPLOYEE (
  `employee_id` INT AUTO_INCREMENT NOT NULL,
  `first_name` VARCHAR(30),
  `last_name` VARCHAR(30),
  `hire_date` DATE,
  `job_title` VARCHAR(30),
  `shop_id` INT,
  PRIMARY KEY (`employee_id`),
  FOREIGN KEY (`shop_id`) REFERENCES
COFFEE_SHOP(`shop_id`)
```

```sql
SELECT * FROM COFFEE_SHOP a
INNER JOIN EMPLOYEE d ON d.shop_id = a.shop_id
INNER JOIN COFFEE o ON o.shop_id = d.shop_id
INNER JOIN SUPPLIER c ON c.supplier_id = o.supplier_id;
```

shop_id	shop_name	city	state	employee_id	first_name	last_name	hire_date	job_title	shop_id	coffee_id	shop_id
1	Bravo	Tustin	CA	1	Jarone	McCorkle	2022-04-12	Software Engineer	1	1	1
2	Echo	Irvine	CA	2	Nancy	Tran	2022-10-12	Receptionist	2	2	2
3	Charlie	Newport	CA	3	Calvin	Klein	2022-09-12	Designer	3	3	3



shop_id	supplier_id	coffee_name	price_per_pound	supplier_id	company_name	country	sales_contact_name	email
1	1	mocha	9.25	1	Balenciaga	Italy	Post Malone	postmalone@gmail.com
2	2	vanilla	4.25	2	Nordstrom	France	Jay Z	jayz@yahoo.com
3	3	decaf	12.52	3	Palace	United States	Kurt Cobain	kurtcobain@proton.me