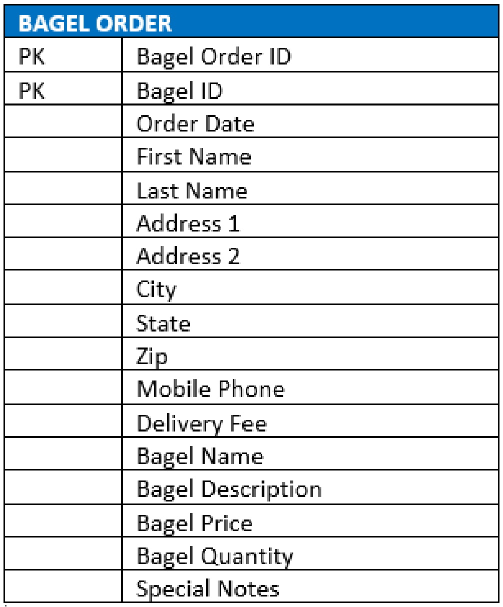
PART A
Nora’s Bagel Bin Database Blueprints

Having been provided the bagel order form for Nora’s Bagel Bin and the first normal form of the database, it was determined that the overall structure of the database could be normalized to enhance functionality and avoid inefficiencies.

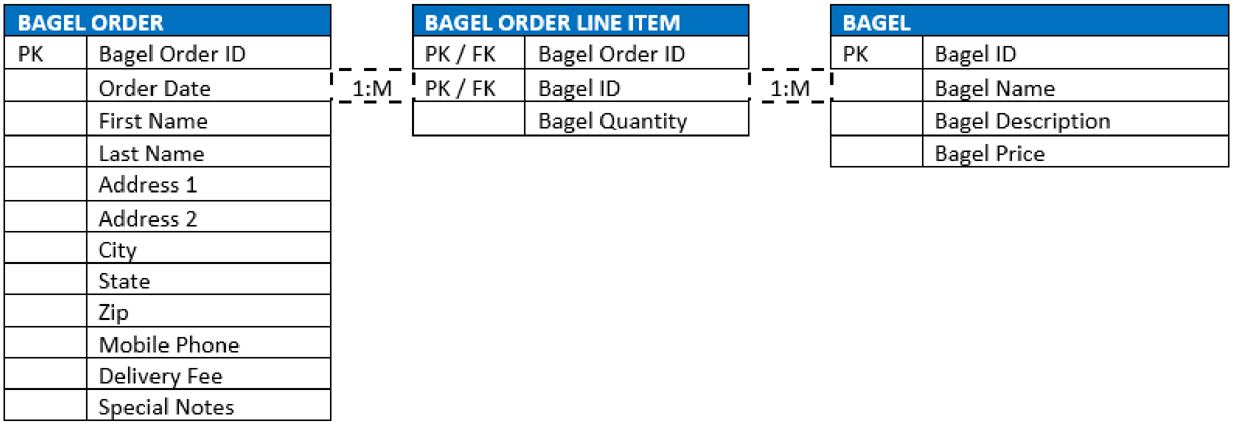
First Normal Form (1NF):



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

In the first normal form, though all of the order form’s fields are captured for further processing, the table containing that data contains a composite key, indicating multiple distinct entities in the table. Because of this, the data is functionally dependent on multiple entities. As such, the database should be normalized further to eliminate data redundancy.

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

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

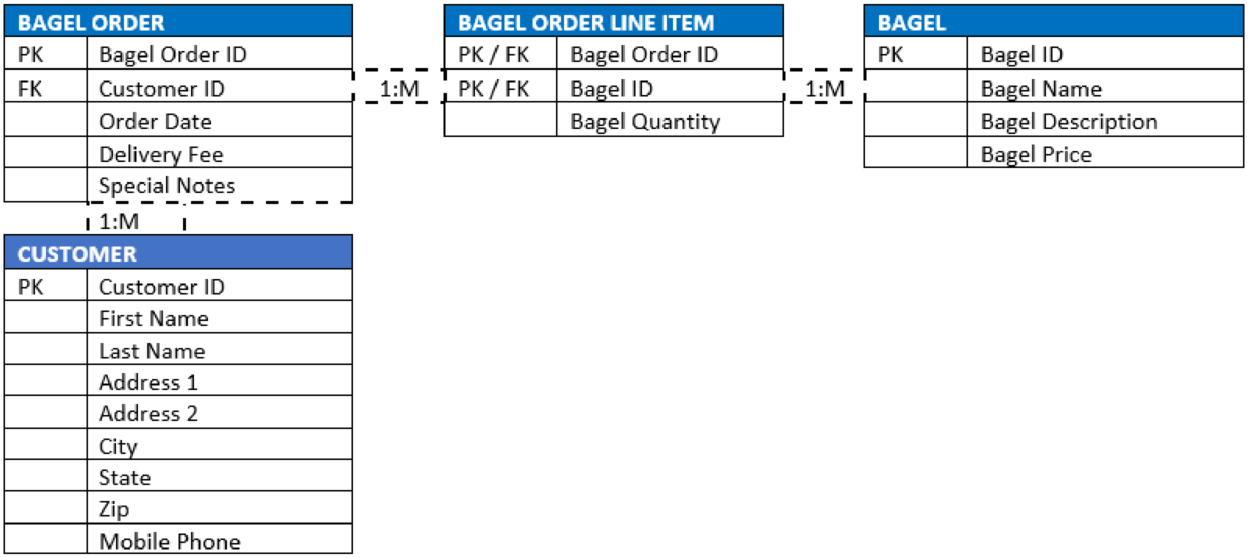
BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

The initial bagel order column has been broken up to contain three entities in the second normal form. The Bagel Order table contains information that will likely be unique to each order submitted to Nora’s (i.e., the order ID, the order date, and the customer’s information), while the information relating to the products offered by Nora’s was placed into the Bagel table; because this information will not change from order to order, it was illogical to keep it within the Bagel Order table. The Bagel Order Line Item acts as a bridge between the Bagel Order and Bagel entities, effectively avoiding the inherent many-to-many relationship between Bagel Orders and Bagels—the Bagel Quantity field exists within this table to ensure that each quantity is dependent on each unique order, while also maintaining a close association to the Bagel entity to which it is related.

The relationships between each entity were determined as follows:

- Each Bagel Order will contain many line items, but each line item can only be associated with one specific order, as it relies on the Bagel Order ID key.
- Similarly, though each Bagel Order Line Item may be associated with multiple types of bagels, the Bagel ID and Bagel Quantity datapoints ensure that each instance of a line item will be independent of the bagels stocked by Nora's.

Third Normal Form (3NF):



BAGEL ORDER	
PK	Bagel Order ID
FK	Customer ID
	Order Date
	Delivery Fee
	Special Notes

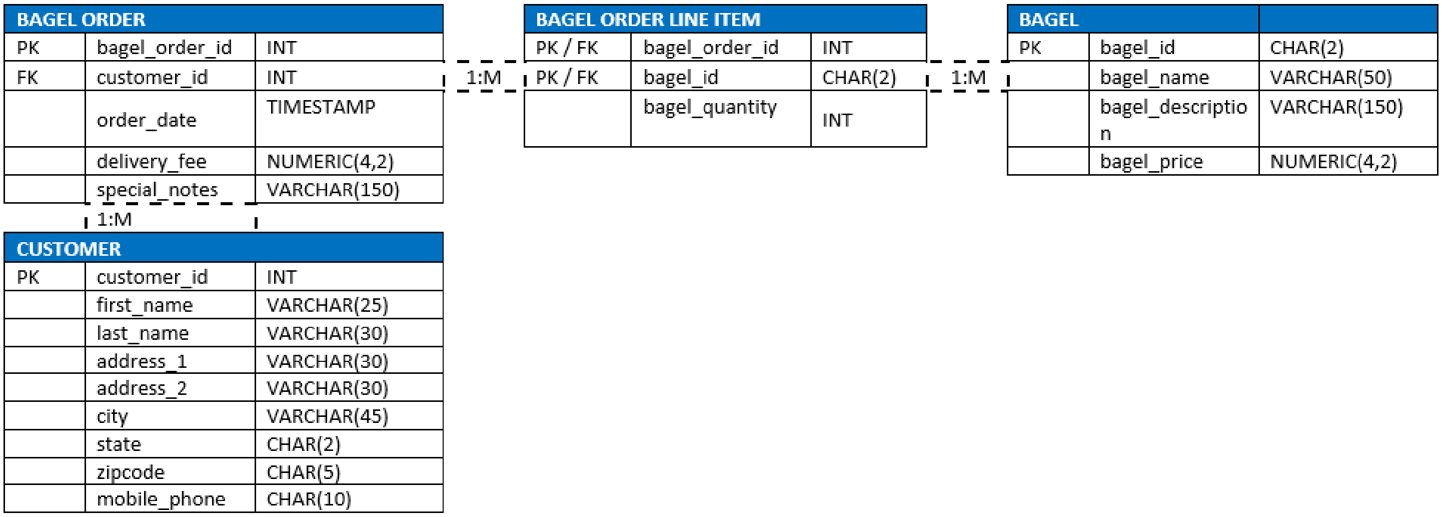
BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Bagel Quantity

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
	Mobile Phone

The changes observed within the third normal form of the database are evident—the entities added in the second normal form still exist, but the Bagel Order entity has been broken down further to account for the separate Customer entity—it is natural that the same customer may place several distinct orders (hinting at the cardinality—namely, one customer may place many orders, but each order will be associated with only one customer). By creating this fourth entity, customers can exist independently of each order, further reducing data redundancy by ensuring customer information is not needlessly duplicated.

Final Physical Database Model:



BAGEL ORDER		
PK	bagel_order_id	INT
FK	customer_id	INT
	order_date	TIMESTAMP
	delivery_fee	NUMERIC(4,2)
	special_notes	VARCHAR(150)

BAGEL ORDER LINE ITEM		
PK / FK	bagel_order_id	INT
PK / FK	bagel_id	CHAR(2)
	bagel_quantity	INT

BAGEL		
PK	bagel_id	CHAR(2)
	bagel_name	VARCHAR(50)
	bagel_description	VARCHAR(150)
	bagel_price	NUMERIC(4,2)

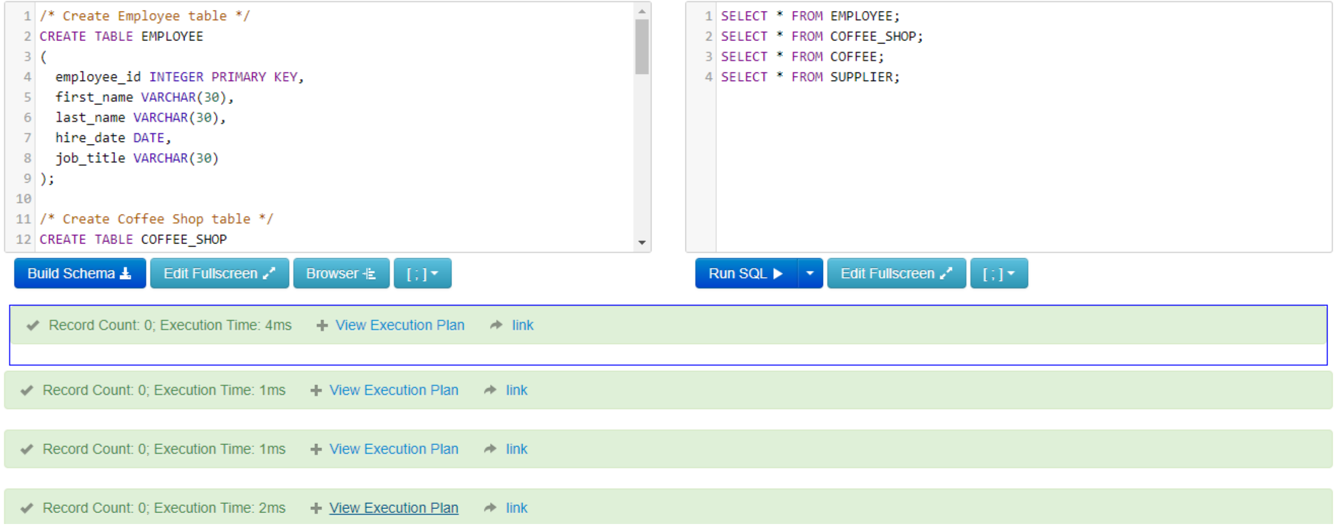
CUSTOMER		
PK	customer_id	INT
	first_name	VARCHAR(25)
	last_name	VARCHAR(30)
	address_1	VARCHAR(30)
	address_2	VARCHAR(30)
	city	VARCHAR(45)
	state	CHAR(2)
	zipcode	CHAR(5)
	mobile_phone	CHAR(10)

This model reflects the changes made in the third normal form, but with appropriate datatypes assigned, and with the names of each field modified to be compatible with SQL's language requirements.

PART B
Jaunty Coffee Co. Database

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

```
/* Create Employee table */
CREATE TABLE EMPLOYEE
(
  employee_id INTEGER PRIMARY KEY,
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  hire_date DATE,
  job_title VARCHAR(30)
);
/* Create Coffee Shop table */
CREATE TABLE COFFEE_SHOP
(
  shop_id INTEGER PRIMARY KEY,
  shop_name VARCHAR(50),
  city VARCHAR(50),
  state CHAR(2)
);
/* Create Coffee table */
CREATE TABLE COFFEE
(
  coffee_id INTEGER PRIMARY KEY,
  coffee_name VARCHAR(30),
  price_per_pound NUMERIC(5,2)
);
/* Create Supplier table */
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
);
/* Add shop_id FK */
ALTER TABLE EMPLOYEE
  ADD shop_id INTEGER,
  ADD FOREIGN KEY(shop_id) REFERENCES COFFEE_SHOP(shop_id);
/* Add shop_id and supplier_id FKs */
ALTER TABLE COFFEE
  ADD shop_id INTEGER,
  ADD supplier_id INTEGER,
  ADD FOREIGN KEY(shop_id) REFERENCES COFFEE_SHOP(shop_id),
  ADD FOREIGN KEY(supplier_id) REFERENCES SUPPLIER(supplier_id);
```

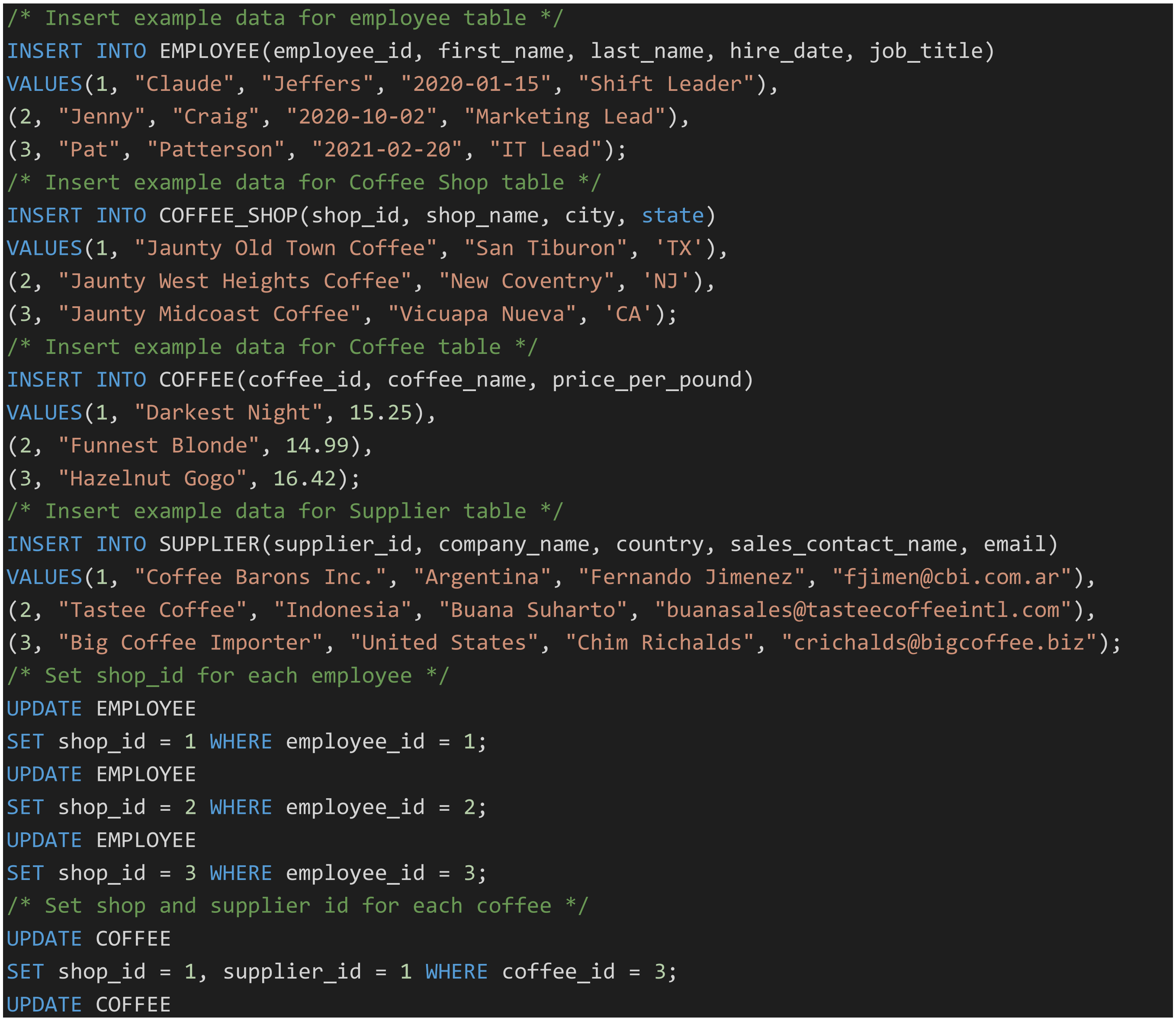


```
/* Create Employee table */
CREATE TABLE EMPLOYEE
(
  employee_id INTEGER PRIMARY KEY,
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  hire_date DATE,
  job_title VARCHAR(30)
);

/* Create Coffee Shop table */
CREATE TABLE COFFEE_SHOP
```

```
SELECT * FROM EMPLOYEE;
SELECT * FROM COFFEE_SHOP;
SELECT * FROM COFFEE;
SELECT * FROM SUPPLIER;
```

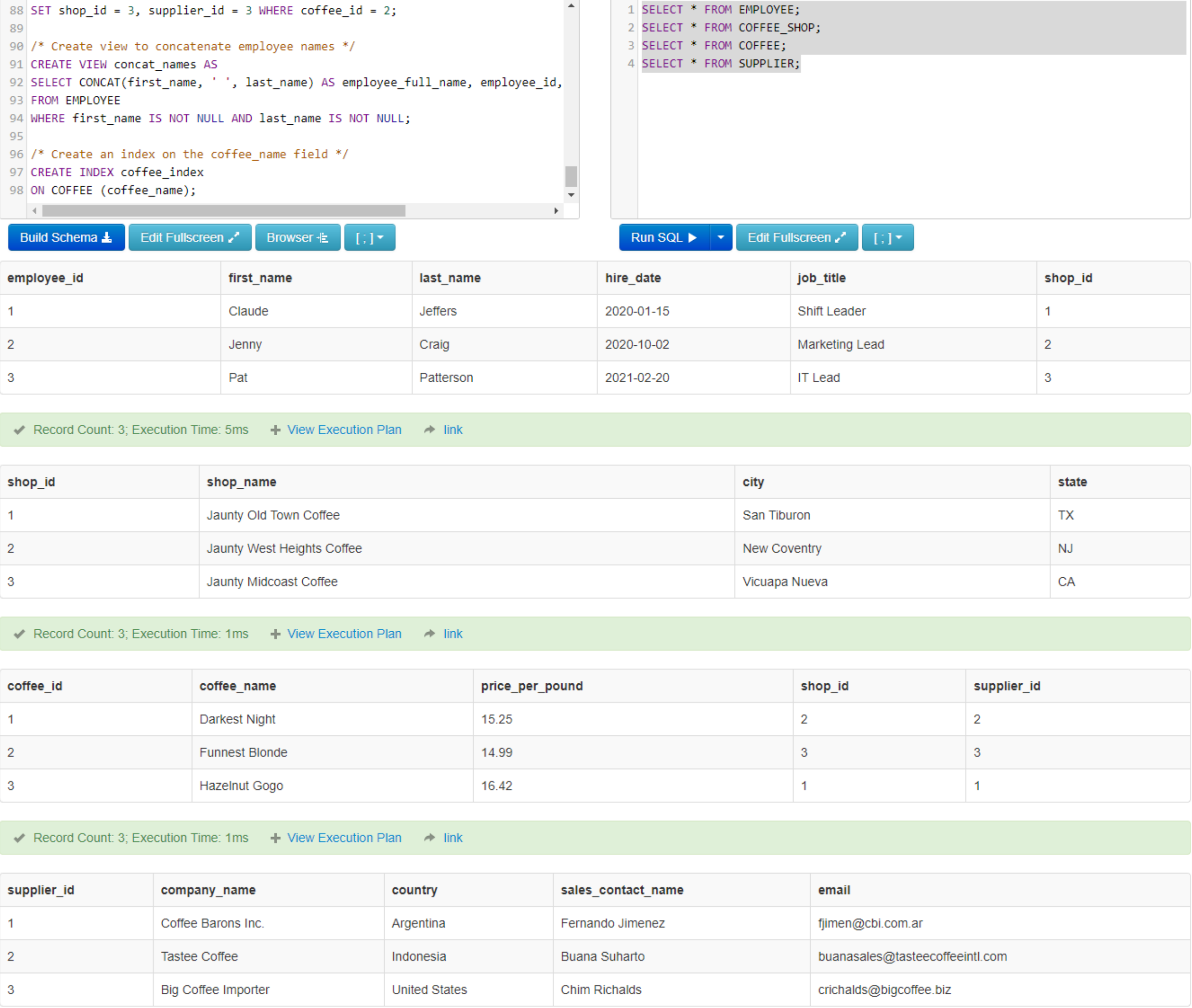
```
✓ Record Count: 0; Execution Time: 4ms   + View Execution Plan   link
✓ Record Count: 0; Execution Time: 1ms   + View Execution Plan   link
✓ Record Count: 0; Execution Time: 1ms   + View Execution Plan   link
✓ Record Count: 0; Execution Time: 2ms   + View Execution Plan   link
```

2. Develop SQL code to populate each table in the database design document



```
/* Insert example data for employee table */
INSERT INTO EMPLOYEE(employee_id, first_name, last_name, hire_date, job_title)
VALUES(1, "Claude", "Jeffers", "2020-01-15", "Shift Leader"),
(2, "Jenny", "Craig", "2020-10-02", "Marketing Lead"),
(3, "Pat", "Patterson", "2021-02-20", "IT Lead");
/* Insert example data for Coffee Shop table */
INSERT INTO COFFEE_SHOP(shop_id, shop_name, city, state)
VALUES(1, "Jaunty Old Town Coffee", "San Tiburon", 'TX'),
(2, "Jaunty West Heights Coffee", "New Coventry", 'NJ'),
(3, "Jaunty Midcoast Coffee", "Vicuapa Nueva", 'CA');
/* Insert example data for Coffee table */
INSERT INTO COFFEE(coffee_id, coffee_name, price_per_pound)
VALUES(1, "Darkest Night", 15.25),
(2, "Funnest Blonde", 14.99),
(3, "Hazelnut Gogo", 16.42);
/* Insert example data for Supplier table */
INSERT INTO SUPPLIER(supplier_id, company_name, country, sales_contact_name, email)
VALUES(1, "Coffee Barons Inc.", "Argentina", "Fernando Jimenez", "fjimen@cbi.com.ar"),
(2, "Tastee Coffee", "Indonesia", "Buana Suharto", "buanasales@tasteecoffeeintl.com"),
(3, "Big Coffee Importer", "United States", "Chim Richalds", "crichalds@bigcoffee.biz");
/* Set shop_id for each employee */
UPDATE EMPLOYEE
SET shop_id = 1 WHERE employee_id = 1;
UPDATE EMPLOYEE
SET shop_id = 2 WHERE employee_id = 2;
UPDATE EMPLOYEE
SET shop_id = 3 WHERE employee_id = 3;
/* Set shop and supplier id for each coffee */
UPDATE COFFEE
SET shop_id = 1, supplier_id = 1 WHERE coffee_id = 3;
UPDATE COFFEE
```

```
SET shop_id = 2, supplier_id = 2 WHERE coffee_id = 1;
UPDATE COFFEE
SET shop_id = 3, supplier_id = 3 WHERE coffee_id = 2;
```



```
SET shop_id = 3, supplier_id = 3 WHERE coffee_id = 2;

/* Create view to concatenate employee names */
CREATE VIEW concat_names AS
SELECT CONCAT(first_name, ' ', last_name) AS employee_full_name, employee_id,
FROM EMPLOYEE
WHERE first_name IS NOT NULL AND last_name IS NOT NULL;

/* Create an index on the coffee_name field */
CREATE INDEX coffee_index
ON COFFEE (coffee_name);
```

```
SELECT * FROM EMPLOYEE;
SELECT * FROM COFFEE_SHOP;
SELECT * FROM COFFEE;
SELECT * FROM SUPPLIER;
```

employee_id	first_name	last_name	hire_date	job_title	shop_id
1	Claude	Jeffers	2020-01-15	Shift Leader	1
2	Jenny	Craig	2020-10-02	Marketing Lead	2
3	Pat	Patterson	2021-02-20	IT Lead	3

✔ Record Count: 3; Execution Time: 5ms ✚ View Execution Plan ➦ link

shop_id	shop_name	city	state
1	Jaunty Old Town Coffee	San Tiburon	TX
2	Jaunty West Heights Coffee	New Coventry	NJ
3	Jaunty Midcoast Coffee	Vicuapa Nueva	CA

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

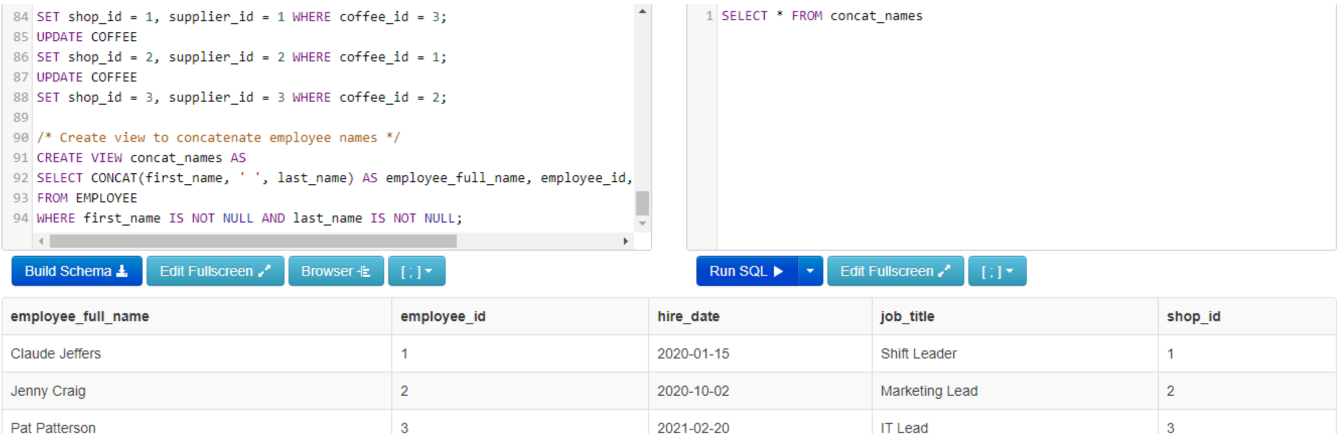
coffee_id	coffee_name	price_per_pound	shop_id	supplier_id
1	Darkest Night	15.25	2	2
2	Funnest Blonde	14.99	3	3
3	Hazelnut Gogo	16.42	1	1

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

supplier_id	company_name	country	sales_contact_name	email
1	Coffee Barons Inc.	Argentina	Fernando Jimenez	fjimen@cbi.com.ar
2	Tastee Coffee	Indonesia	Buana Suharto	buanasales@tasteecoffeeintl.com
3	Big Coffee Importer	United States	Chim Richalds	crichalds@bigcoffee.biz

3. **Develop SQL code to create a view showing all information from the EMPLOYEE table, with the new employee_full_name attribute**

```
/* Create view to concatenate employee names */
CREATE VIEW concat_names AS
SELECT CONCAT(first_name, ' ', last_name) AS employee_full_name, employee_id, hire_date,
job_title, shop_id
FROM EMPLOYEE
WHERE first_name IS NOT NULL AND last_name IS NOT NULL;
```

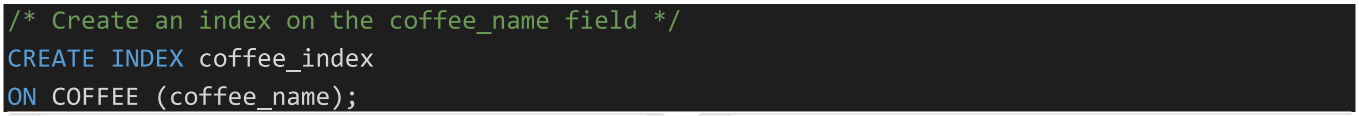


```sql
SET shop_id = 1, supplier_id = 1 WHERE coffee_id = 3;
UPDATE COFFEE
SET shop_id = 2, supplier_id = 2 WHERE coffee_id = 1;
UPDATE COFFEE
SET shop_id = 3, supplier_id = 3 WHERE coffee_id = 2;

/* Create view to concatenate employee names */
CREATE VIEW concat_names AS
SELECT CONCAT(first_name, ' ', last_name) AS employee_full_name, employee_id,
FROM EMPLOYEE
WHERE first_name IS NOT NULL AND last_name IS NOT NULL;
```

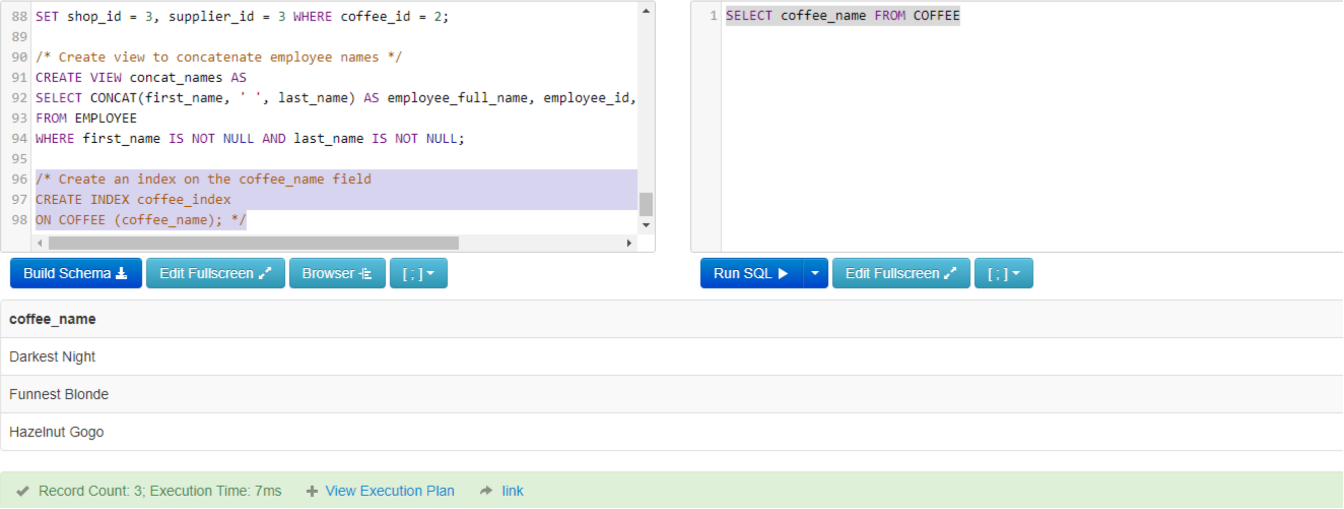
```sql
SELECT * FROM concat_names
```

employee_full_name	employee_id	hire_date	job_title	shop_id
Claude Jeffers	1	2020-01-15	Shift Leader	1
Jenny Craig	2	2020-10-02	Marketing Lead	2
Pat Patterson	3	2021-02-20	IT Lead	3

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



```sql
/* Create an index on the coffee_name field */
CREATE INDEX coffee_index
ON COFFEE (coffee_name);
```

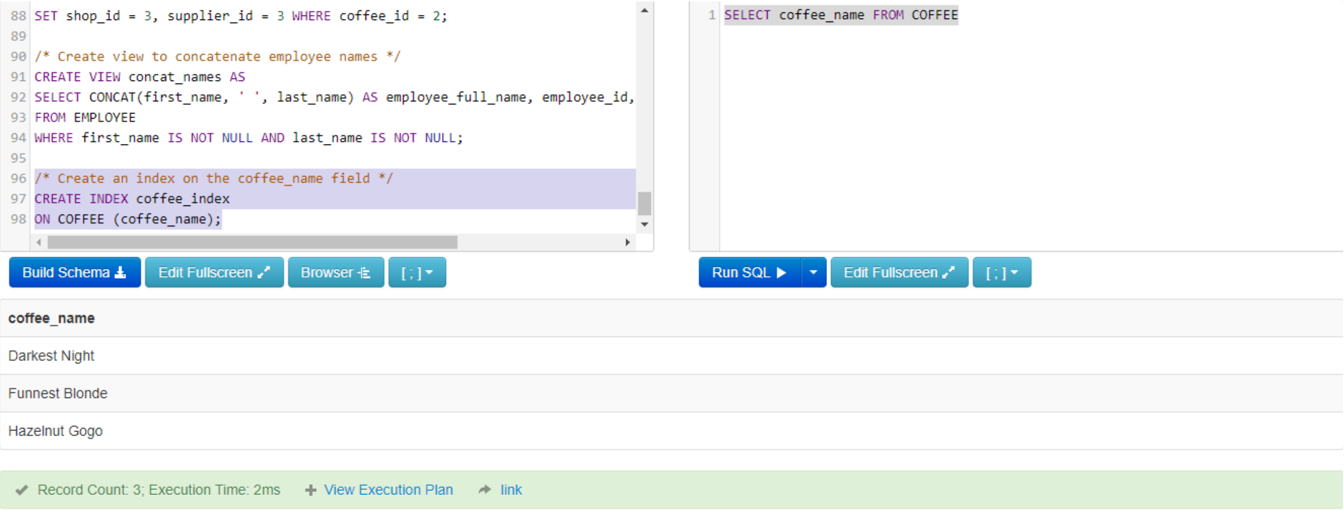


```sql
SET shop_id = 3, supplier_id = 3 WHERE coffee_id = 2;

/* Create view to concatenate employee names */
CREATE VIEW concat_names AS
SELECT CONCAT(first_name, ' ', last_name) AS employee_full_name, employee_id,
FROM EMPLOYEE
WHERE first_name IS NOT NULL AND last_name IS NOT NULL;

/* Create an index on the coffee_name field
CREATE INDEX coffee_index
ON COFFEE (coffee_name); */
```

```sql
SELECT coffee_name FROM COFFEE
```

coffee_name
Darkest Night
Funnest Blonde
Hazelnut Gogo

✔ Record Count: 3; Execution Time: 7ms + View Execution Plan ➦ link



```sql
SET shop_id = 3, supplier_id = 3 WHERE coffee_id = 2;

/* Create view to concatenate employee names */
CREATE VIEW concat_names AS
SELECT CONCAT(first_name, ' ', last_name) AS employee_full_name, employee_id,
FROM EMPLOYEE
WHERE first_name IS NOT NULL AND last_name IS NOT NULL;

/* Create an index on the coffee_name field */
CREATE INDEX coffee_index
ON COFFEE (coffee_name);
```

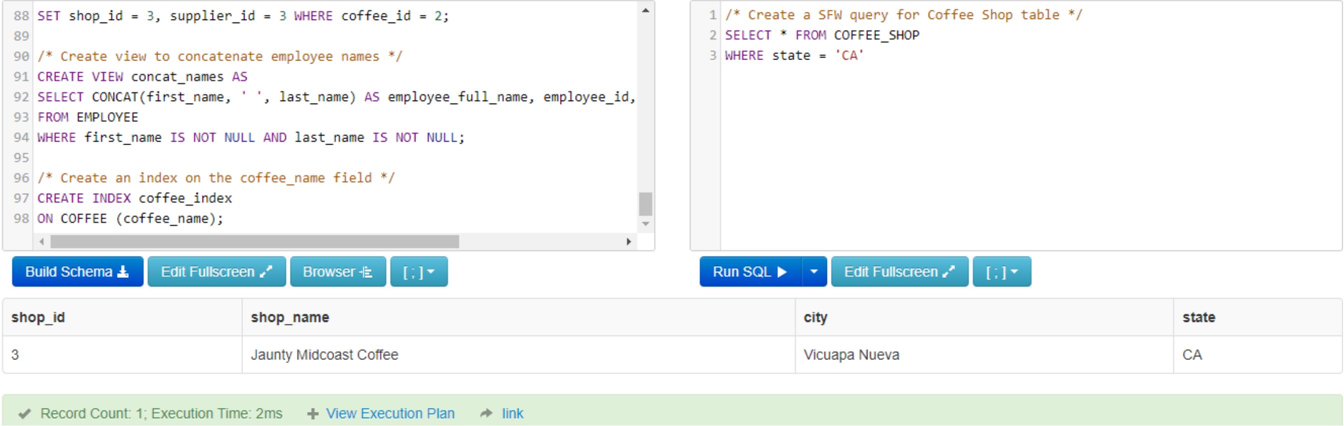
```sql
SELECT coffee_name FROM COFFEE
```

coffee_name
Darkest Night
Funnest Blonde
Hazelnut Gogo

✔ Record Count: 3; Execution Time: 2ms + View Execution Plan ➦ link

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

```sql
/* Create a SFW query for Coffee Shop table */
SELECT * FROM COFFEE_SHOP
WHERE state = 'CA'
```



```sql
SET shop_id = 3, supplier_id = 3 WHERE coffee_id = 2;

/* Create view to concatenate employee names */
CREATE VIEW concat_names AS
SELECT CONCAT(first_name, ' ', last_name) AS employee_full_name, employee_id,
FROM EMPLOYEE
WHERE first_name IS NOT NULL AND last_name IS NOT NULL;

/* Create an index on the coffee_name field */
CREATE INDEX coffee_index
ON COFFEE (coffee_name);
```

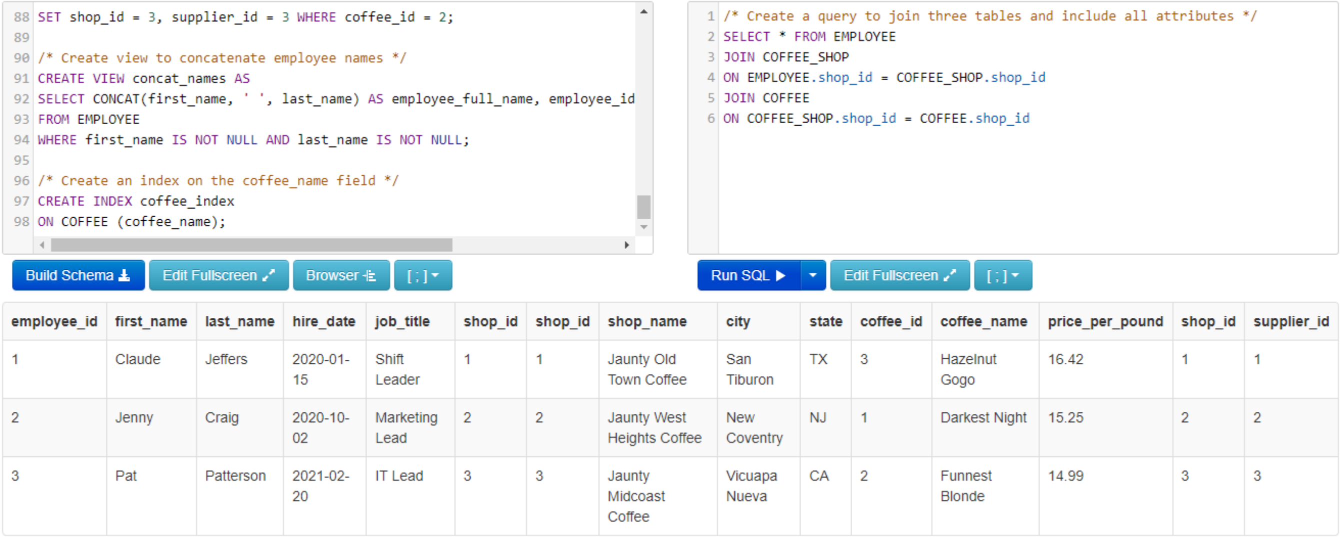
```sql
/* Create a SFW query for Coffee Shop table */
SELECT * FROM COFFEE_SHOP
WHERE state = 'CA'
```

shop_id	shop_name	city	state
3	Jaunty Midcoast Coffee	Vicuapa Nueva	CA

✓ Record Count: 1; Execution Time: 2ms

6. **Develop SQL code to create a query joining three different tables, including attributes from all three**

```sql
/* Create a query to join three tables and include all attributes */
SELECT * FROM EMPLOYEE
JOIN COFFEE_SHOP
ON EMPLOYEE.shop_id = COFFEE_SHOP.shop_id
JOIN COFFEE
ON COFFEE_SHOP.shop_id = COFFEE.shop_id
```



```sql
SET shop_id = 3, supplier_id = 3 WHERE coffee_id = 2;

/* Create view to concatenate employee names */
CREATE VIEW concat_names AS
SELECT CONCAT(first_name, ' ', last_name) AS employee_full_name, employee_id
FROM EMPLOYEE
WHERE first_name IS NOT NULL AND last_name IS NOT NULL;

/* Create an index on the coffee_name field */
CREATE INDEX coffee_index
ON COFFEE (coffee_name);
```

```sql
/* Create a query to join three tables and include all attributes */
SELECT * FROM EMPLOYEE
JOIN COFFEE_SHOP
ON EMPLOYEE.shop_id = COFFEE_SHOP.shop_id
JOIN COFFEE
ON COFFEE_SHOP.shop_id = COFFEE.shop_id
```

employee_id	first_name	last_name	hire_date	job_title	shop_id	shop_id	shop_name	city	state	coffee_id	coffee_name	price_per_pound	shop_id	supplier_id
1	Claude	Jeffers	2020-01-15	Shift Leader	1	1	Jaunty Old Town Coffee	San Tiburon	TX	3	Hazelnut Gogo	16.42	1	1
2	Jenny	Craig	2020-10-02	Marketing Lead	2	2	Jaunty West Heights Coffee	New Coventry	NJ	1	Darkest Night	15.25	2	2
3	Pat	Patterson	2021-02-20	IT Lead	3	3	Jaunty Midcoast Coffee	Vicuapa Nueva	CA	2	Funnest Blonde	14.99	3	3