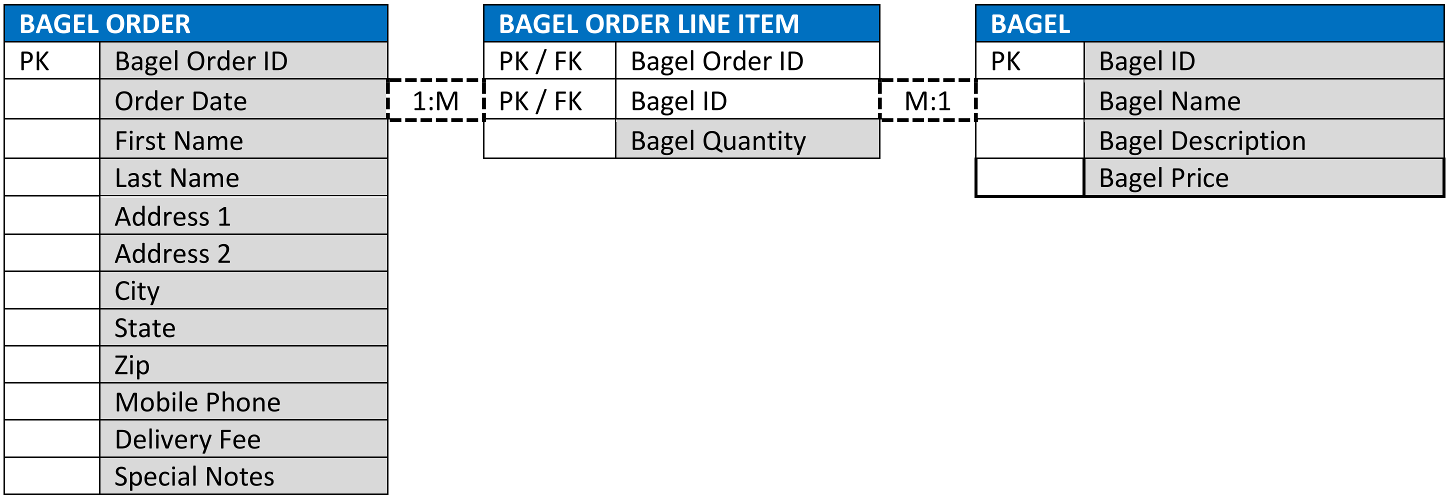
Data Management Applications – C170
Matthew Scillitani

Part A: Nora’s Bagel Bin

1. 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

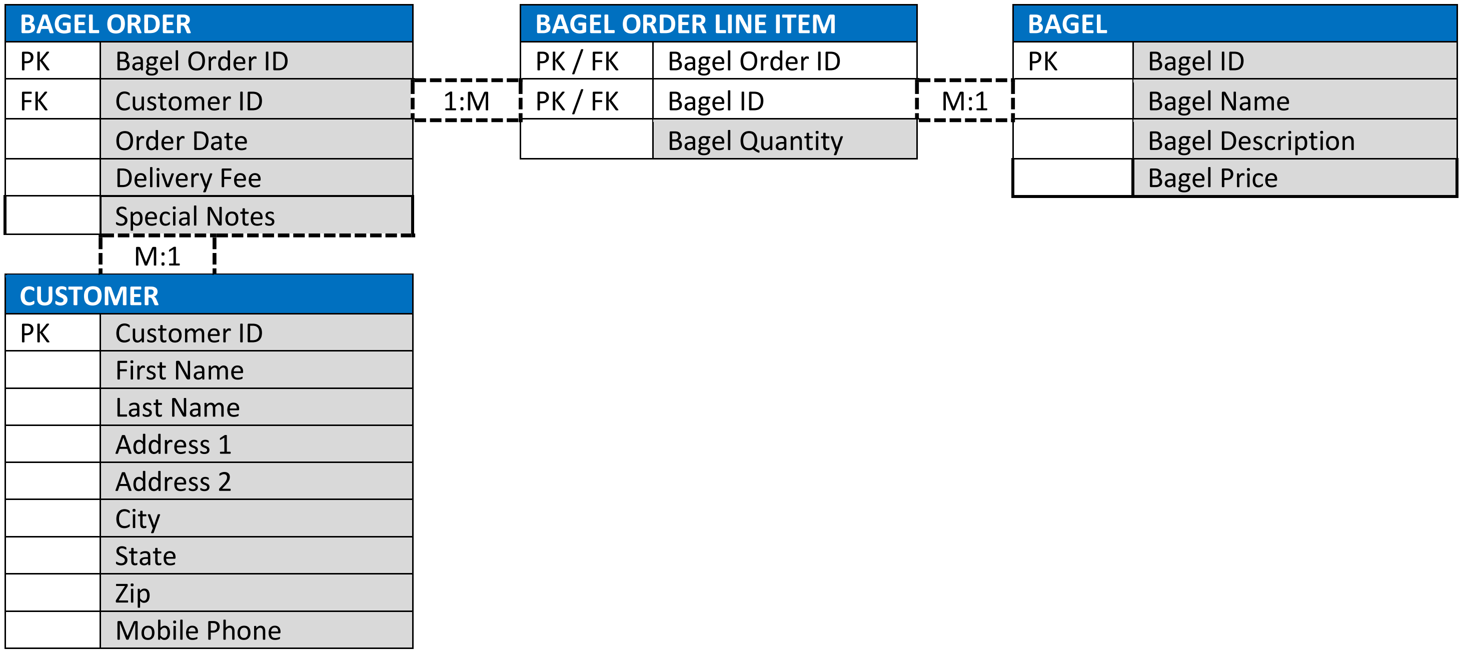
Attributes explanation:

For every table, attributes were assigned based on relevance and uniqueness. The BAGEL ORDER table is a container for all order and customer information, the BAGEL ORDER LINE ITEM table holds all the information for every line item, and the BAGEL table holds all the information about every bagel.

Cardinality explanation:

The BAGEL ORDER and BAGEL ORDER LINE ITEM tables have a 1:M relationship because every bagel order may have many line items, but each line item will be unique to an order. The BAGEL ORDER LINE ITEM and BAGEL tables have a M:1 relationship because every line item has only one bagel and many bagels will be used as line items.

2. 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

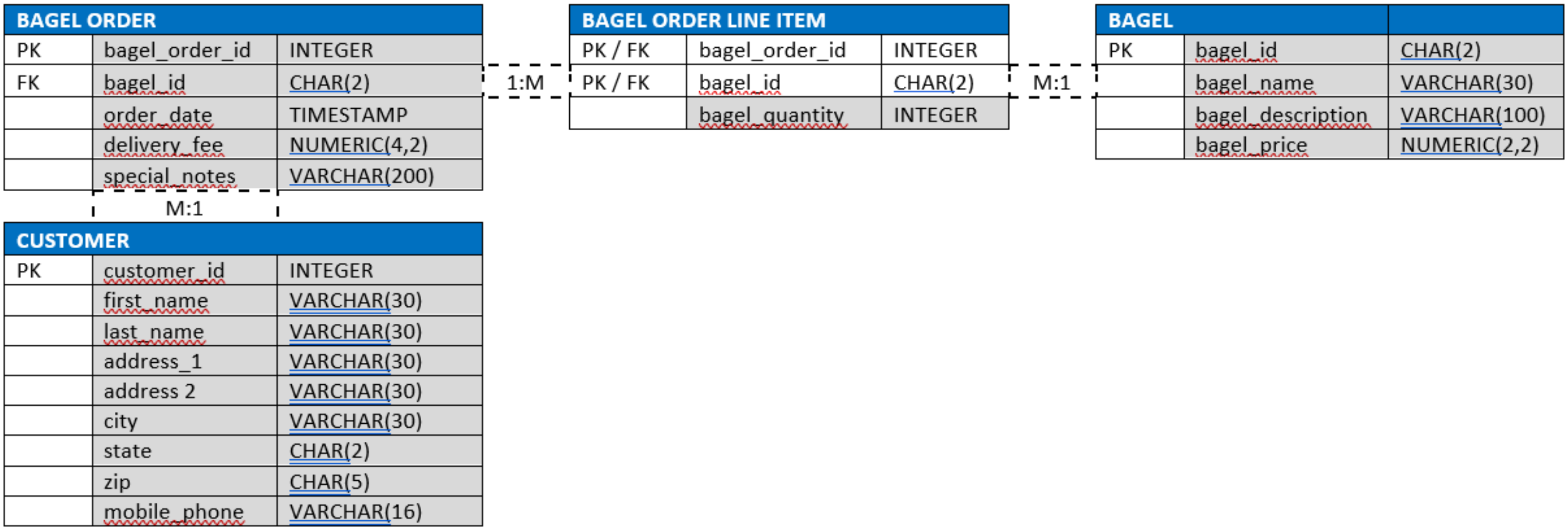
Attributes explanation:

Attributes were chosen by relevance and uniqueness, so that the CUSTOMER table contains exclusively customer information, the BAGEL ORDER table contains exclusively the information that is necessary for completing orders, the BAGEL ORDER LINE ITEM table contains only the information comprising a single line in the order (such as 12 plain bagels), and the BAGEL table contains only relevant bagel information.

Cardinality explanation:

There is a M:1 relationship between BAGEL ORDER and CUSTOMER because many orders can be made by one customer and orders are unique. There is a 1:M relationship between BAGEL ORDER and BAGEL ORDER ITEM because one bagel order may have many line items and every line item is unique to an order. The relationship between BAGEL LINE ITEM and BAGELS is M:1 because line items contain only one type of bagel, and many bagels are used for different line items.

3. Final physical database model



BAGEL ORDER		
PK	bagel_order_id	INTEGER
FK	bagel_id	CHAR(2)
	order_date	TIMESTAMP
	delivery_fee	NUMERIC(4,2)
	special_notes	VARCHAR(200)

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

BAGEL		
PK	bagel_id	CHAR(2)
	bagel_name	VARCHAR(30)
	bagel_description	VARCHAR(100)
	bagel_price	NUMERIC(2,2)

CUSTOMER		
PK	customer_id	INTEGER
	first_name	VARCHAR(30)
	last_name	VARCHAR(30)
	address_1	VARCHAR(30)
	address 2	VARCHAR(30)
	city	VARCHAR(30)
	state	CHAR(2)
	zip	CHAR(5)
	mobile_phone	VARCHAR(16)

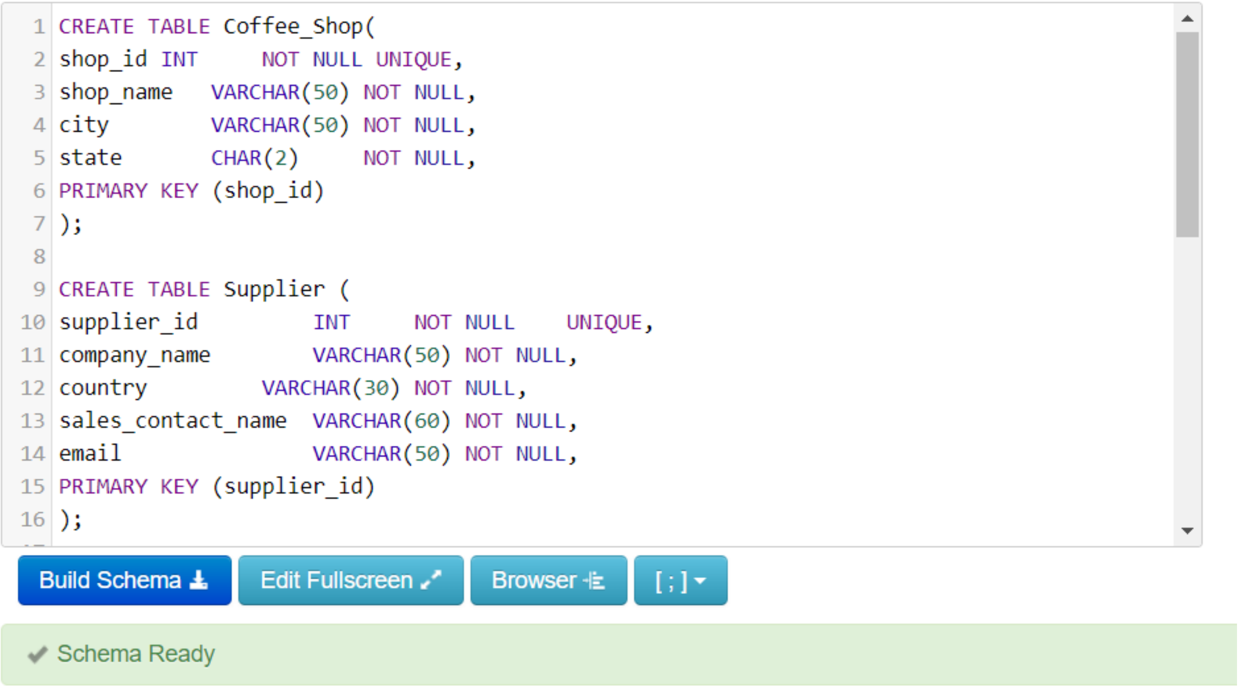
Part B: Jaunty Coffee Co. ERD

1. SQL code

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

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

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

CREATE TABLE Coffee (
    coffee_id INT NOT NULL UNIQUE,
    shop_id INT,
    supplier_id INT,
    coffee_name VARCHAR(30) NOT NULL,
    price_per_pound NUMERIC(5, 2) NOT NULL,
    PRIMARY KEY (coffee_id),
    FOREIGN KEY (shop_id) REFERENCES Coffee_Shop (shop_id),
    FOREIGN KEY (supplier_id) REFERENCES Supplier (supplier_id)
);
```

Screenshot showing the successful execution of the above code:



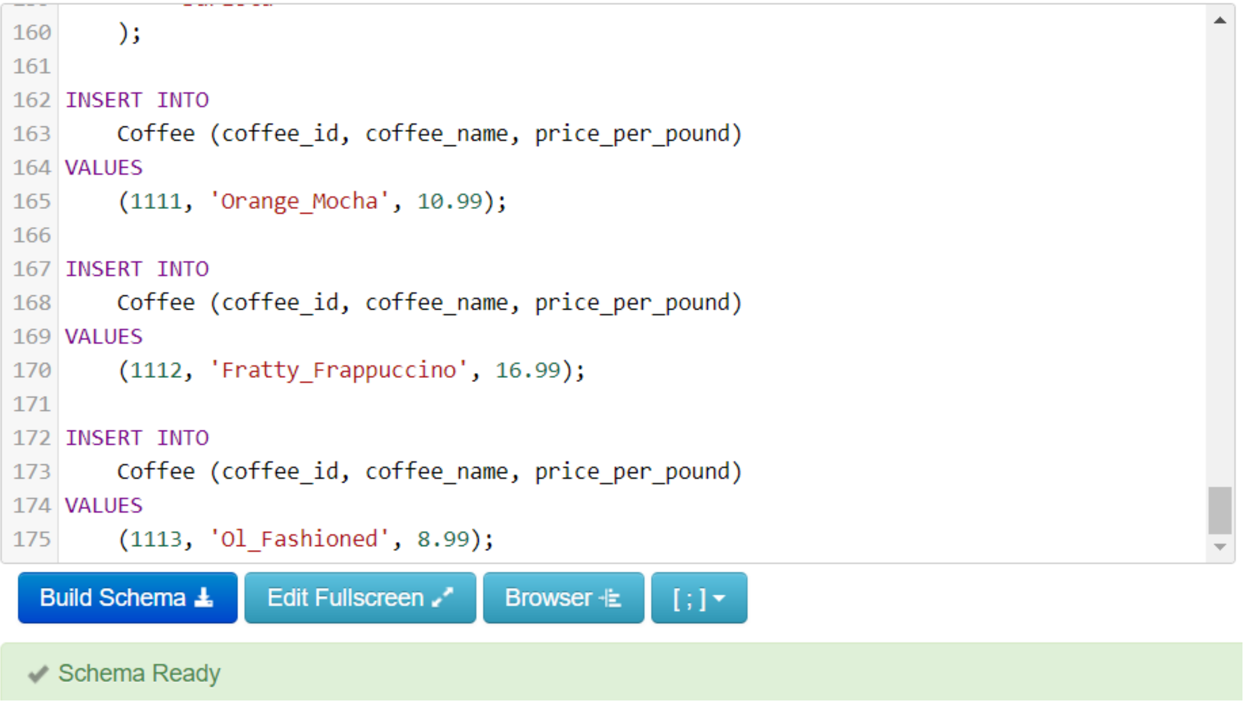
```
CREATE TABLE Coffee_Shop(
shop_id INT     NOT NULL UNIQUE,
shop_name   VARCHAR(50) NOT NULL,
city        VARCHAR(50) NOT NULL,
state       CHAR(2)     NOT NULL,
PRIMARY KEY (shop_id)
);

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

2. SQL code populating each table

```
INSERT INTO
    Coffee_Shop (shop_id, shop_name, city, state)
VALUES
    (1, 'RatsBucks', 'San_Francisco', 'CA');

INSERT INTO
    Coffee_Shop (shop_id, shop_name, city, state)
VALUES
    (2, 'Mongoose_Coffee', 'Raleigh', 'NC');

INSERT INTO
    Coffee_Shop (shop_id, shop_name, city, state)
VALUES
    (
        3,
        'Dons_Covfefe',
        'Dallas',
        'TX'
    );

INSERT INTO
    Supplier (
        supplier_id,
        company_name,
        country,
        sales_contact_name,
        email
    )
VALUES
    (
        11,
        'Biggest_Beans_Co',
        'America',
        'Donald',
        'BigDon@gmail.com'
    );

INSERT INTO
    Supplier (
        supplier_id,
        company_name,
        country,
        sales_contact_name,
        email
    )
VALUES
    (
        12,
        'Frat_Frap_Inc',
        'America',
        'Chad',
        'Chadwick@gmail.com'
    );

INSERT INTO
    Supplier (
        supplier_id,
        company_name,
        country,
        sales_contact_name,
        email
    )
VALUES
    (
        13,
        'Crows_Crawfee_LLC',
        'England',
        'Allister',
        'MrCrowley@gmail.com'
    );

INSERT INTO
    Employee (
        employee_id,
        first_name,
        last_name,
        hire_date,
        job_title
    )
VALUES
    (
        111,
        'Chris',
        'Pinecone',
        '2020-12-11',
        'Manager'
    );

INSERT INTO
    Employee (
        employee_id,
        first_name,
        last_name,
        hire_date,
        job_title
    )
VALUES
    (
        112,
        'Chris',
        'Evens',
        '2017-10-21',
        'Accountant'
    );

INSERT INTO
    Employee (
        employee_id,
        first_name,
        last_name,
        hire_date,
        job_title
    )
VALUES
    (
        113,
        'Chris',
        'Prattle',
        '2005-10-18',
        'Barista'
    );

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1111, 'Orange_Mocha', 10.99);

INSERT INTO
```

```
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1112, 'Fratty_Frappuccino', 16.99);

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1113, 'Ol_Fashioned', 8.99);
```

Screenshot showing the successful execution of the above code:



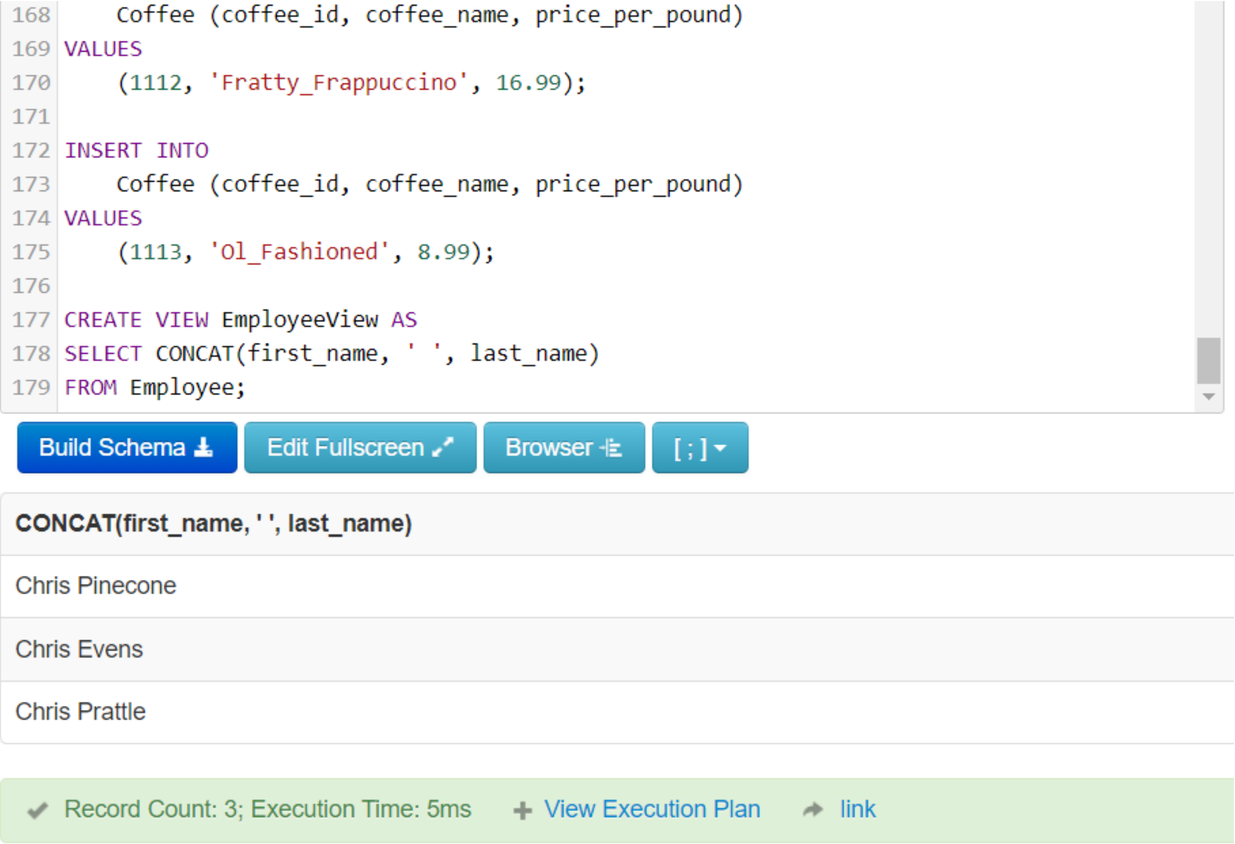
```
    );

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1111, 'Orange_Mocha', 10.99);

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1112, 'Fratty_Frappuccino', 16.99);

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1113, 'Ol_Fashioned', 8.99);
```

✓ Schema Ready

3. SQL code to create a view of employee names

```
CREATE VIEW EmployeeView AS
SELECT CONCAT(first_name, ' ', last_name)
FROM Employee;

SELECT * FROM EmployeeView
```

Screenshot showing the successful execution of the above code:



```
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1112, 'Fratty_Frappuccino', 16.99);

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1113, 'Ol_Fashioned', 8.99);

CREATE VIEW EmployeeView AS
SELECT CONCAT(first_name, ' ', last_name)
FROM Employee;
```

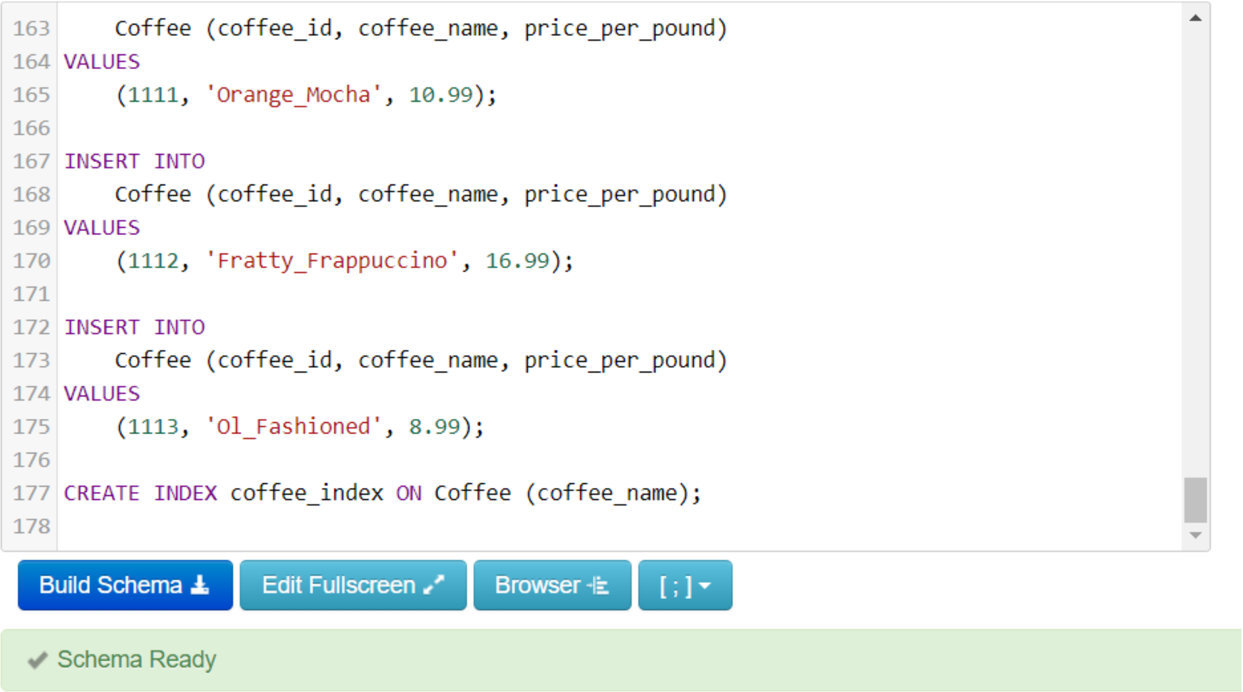
CONCAT(first_name, ' ', last_name)
Chris Pinecone
Chris Evens
Chris Prattle

```
✔ Record Count: 3; Execution Time: 5ms
```

4. SQL code to create an index on the coffee_name field

CREATE INDEX coffee_index ON Coffee (coffee_name);

Screenshot showing the successful execution of the above code:



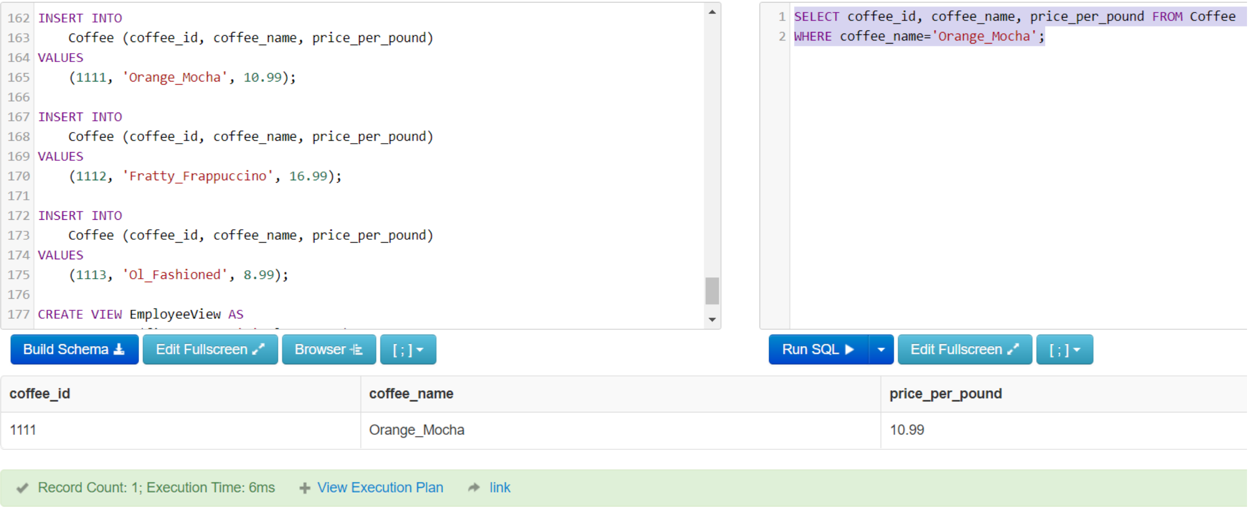
```
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1111, 'Orange_Mocha', 10.99);

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1112, 'Fratty_Frappuccino', 16.99);

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1113, 'Ol_Fashioned', 8.99);

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

✓ Schema Ready

5. SQL code to create an SFW query

SELECT coffee_id, coffee_name, price_per_pound FROM Coffee
WHERE coffee_name='Orange_Mocha';

Screenshot showing the successful execution of the above code:



```
INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1111, 'Orange_Mocha', 10.99);

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1112, 'Fratty_Frappuccino', 16.99);

INSERT INTO
    Coffee (coffee_id, coffee_name, price_per_pound)
VALUES
    (1113, 'Ol_Fashioned', 8.99);

CREATE VIEW EmployeeView AS
```

```
SELECT coffee_id, coffee_name, price_per_pound FROM Coffee
WHERE coffee_name='Orange_Mocha';
```

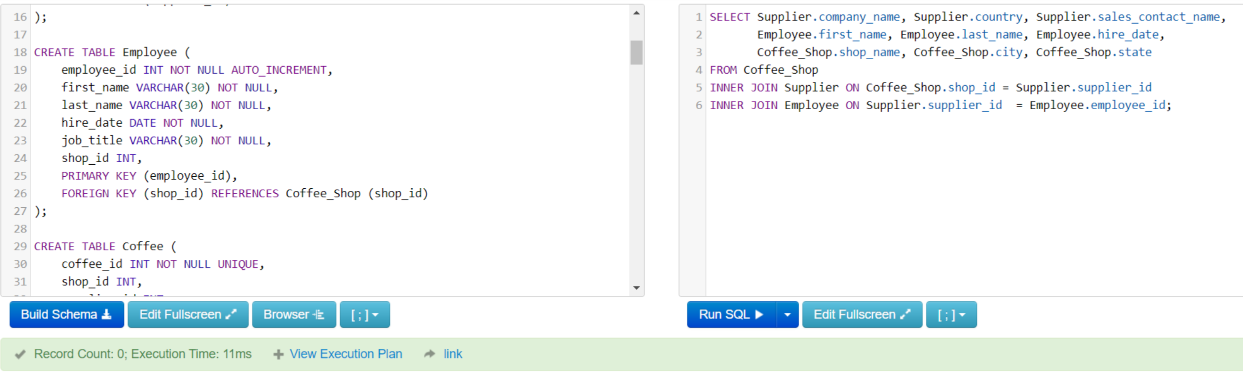
coffee_id	coffee_name	price_per_pound
1111	Orange_Mocha	10.99

✓ Record Count: 1; Execution Time: 6ms

6. SQL code to create a query

```
SELECT Supplier.company_name, Supplier.country,
Supplier.sales_contact_name,
       Employee.first_name, Employee.last_name, Employee.hire_date,
       Coffee_Shop.shop_name, Coffee_Shop.city, Coffee_Shop.state
FROM Coffee_Shop
INNER JOIN Supplier ON Coffee_Shop.shop_id = Supplier.supplier_id
INNER JOIN Employee ON Supplier.supplier_id  = Employee.employee_id;
```

Screenshot showing the successful execution of the above code:



```
);

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

CREATE TABLE Coffee (
    coffee_id INT NOT NULL UNIQUE,
    shop_id INT,
```

```
SELECT Supplier.company_name, Supplier.country, Supplier.sales_contact_name,
       Employee.first_name, Employee.last_name, Employee.hire_date,
       Coffee_Shop.shop_name, Coffee_Shop.city, Coffee_Shop.state
FROM Coffee_Shop
INNER JOIN Supplier ON Coffee_Shop.shop_id = Supplier.supplier_id
INNER JOIN Employee ON Supplier.supplier_id  = Employee.employee_id;
```

✓ Record Count: 0; Execution Time: 11ms