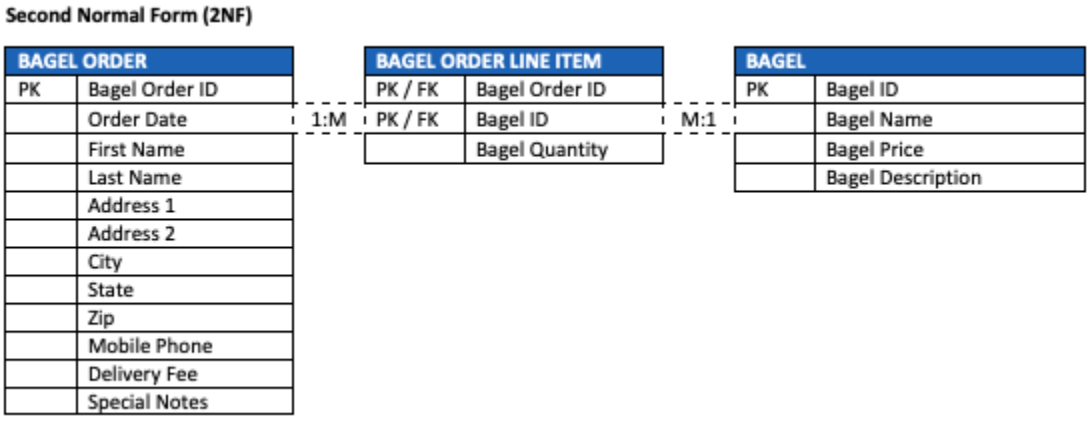
SECTION A

First Normal Form

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

QUESTION 1

Second Normal Form (2NF)



EXPLANATION

The database model for Nora's Bagel Bin was meticulously designed, with a focus on organizing attributes through similarity. The key objective was to create an efficient and normalized structure to represent the ordering process accurately.

To achieve this, the attributes related to bagels, particularly the specific type of bagel and its unique characteristics, were thoughtfully grouped together in the Bagel table. This table holds crucial information such as the bagel's ID, name, and price, making it a comprehensive repository for all bagel-related details.

On the other hand, attributes dealing with the specifics of the order, including customer information, order ID, and order date, were intelligently clustered in the Bagel Order table. This table serves as a central entity capturing essential data for each individual order placed at the shop.

To connect the Bagel Order and Bagel tables seamlessly, a junction table called Bagel Order Line Item was introduced. This junction table facilitates the association between an order and the specific bagels ordered within that order. Each row in the Bagel Order Line Item table corresponds to a distinct bagel ordered, ensuring the accurate linkage between the order and its constituent bagels.

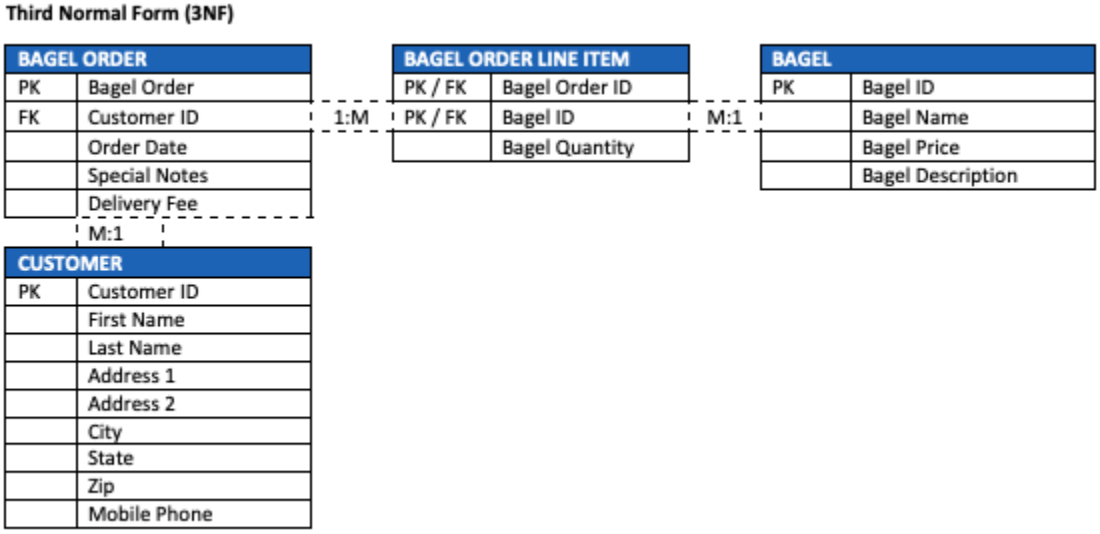
The cardinality between the Bagel Order and Bagel Order Line Item tables was rightfully declared as one-to-many. This is because for each Bagel Order, multiple bagels may be ordered in a single transaction. Consequently, each type of bagel ordered in an order is recorded as an individual row within the Bagel Order Line Item table.

Furthermore, the cardinality between the Bagel Order Line Item and Bagel tables was determined to be many-to-one. This conclusion was drawn based on the fact that a bagel order line item can encompass several different bagels for a single order. Yet, each of these bagels is intrinsically linked to one corresponding entry in the Bagel table, ensuring a consistent and accurate representation of the bagels offered by the coffee shop.

By adhering to these cardinality designations and carefully organizing the attributes into relevant tables, the normalized physical database model for Nora's Bagel Bin achieves a well-structured and efficient representation of the ordering process. This database design not only minimizes redundancy and data inconsistencies but also provides a solid foundation for seamless data retrieval and maintenance, supporting the coffee shop's operations effectively and professionally.

QUESTION 2

Third Normal Form (3NF)



EXPLANATION

To achieve a higher level of normalization and optimize the database structure, further decomposition of the previous table was undertaken, resulting in the creation of additional tables. One of the primary objectives was to break down the Bagel Order table, leading to the introduction of the Customer table to manage customer-related information effectively. This new approach ensures that the Bagel Order table solely contains data relevant to each specific order, while the Customer table is responsible for maintaining customer-related details.

This new design proves to be particularly advantageous for recurring customers as they are assigned a unique Customer ID primary key, which can be utilized as a foreign key in the Bagel Order table. By doing so, there is no need to redundantly list their information in every order they place, leading to a more efficient and streamlined data representation.

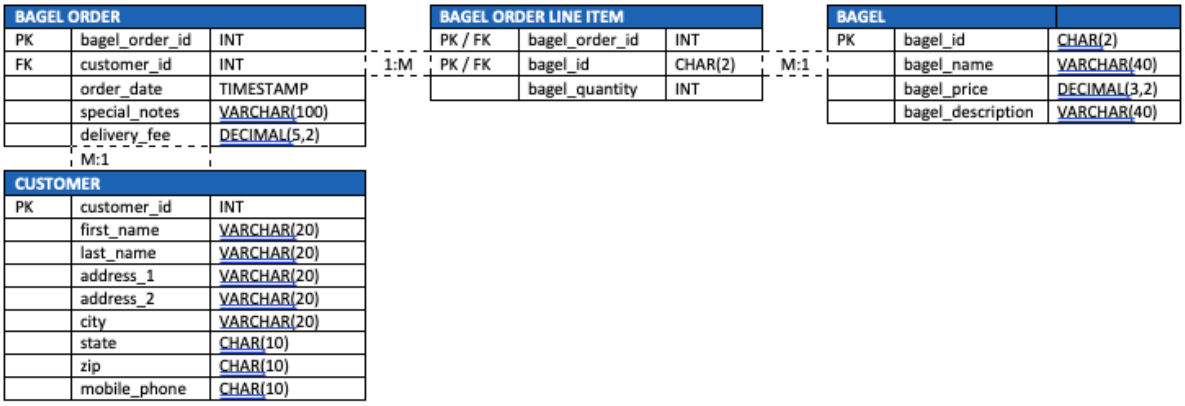
One crucial cardinality relationship was introduced with the application of the third normal form. This cardinality exists between the Bagel Order and Customer tables and is classified as many-to-one. This relationship arises because a single customer can be associated with multiple orders over time. For regular customers, this could translate to a substantial number of orders, while their core information, such as name and contact details, remains constant. Consequently, their unique Customer ID serves as a stable reference, simplifying data maintenance and ensuring consistency throughout the database.

In conclusion, the decision to further normalize the database by employing the third normal form and introducing additional tables has proven to be highly beneficial. This approach allows for a more organized and efficient representation of data, minimizing redundancy and enhancing data integrity. The establishment of a distinct Customer table enables seamless management of customer information, while the streamlined Bagel Order table optimizes the representation of order-specific data. By adhering to the many-to-one cardinality relationship between the Bagel Order and Customer tables, the database design ensures accurate and structured storage of customer and order-related information, contributing to a robust and professional database solution.

QUESTION 3

Final Physical Database Model

Final Physical Database Model



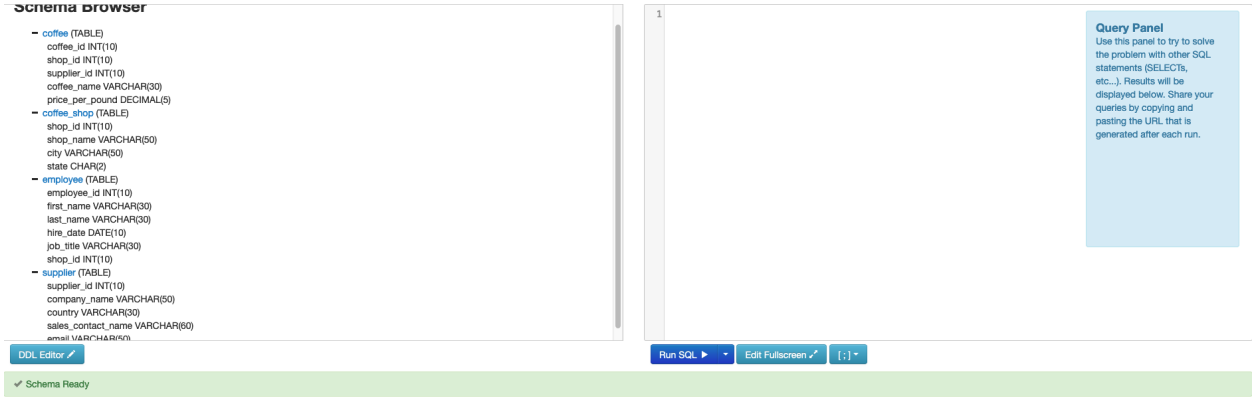
SECTION B

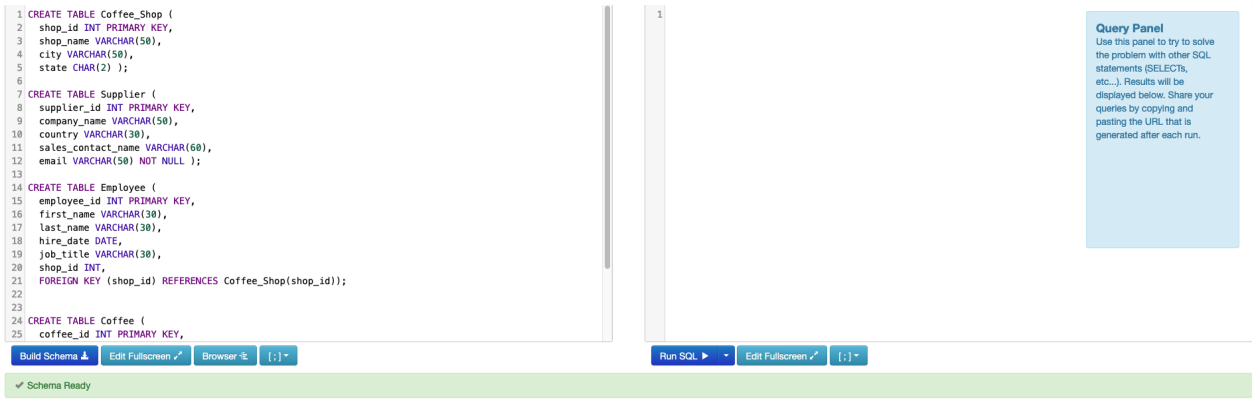
QUESTION 1

A

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

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 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 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));
```

B


Schema Browser
coffee (TABLE)
coffee_id INT(10)
shop_id INT(10)
supplier_id INT(10)
coffee_name VARCHAR(30)
price_per_pound DECIMAL(5)
coffee_shop (TABLE)
shop_id INT(10)
shop_name VARCHAR(50)
city VARCHAR(50)
state CHAR(2)
employee (TABLE)
employee_id INT(10)
first_name VARCHAR(30)
last_name VARCHAR(30)
hire_date DATE(10)
job_title VARCHAR(30)
shop_id INT(10)
supplier (TABLE)
supplier_id INT(10)
company_name VARCHAR(50)
country VARCHAR(30)
sales_contact_name VARCHAR(60)
DDL Editor
Run SQL
Edit Fullscreen
[;]
Query Panel
Use this panel to try to solve the problem with other SQL statements (SELECTs, etc...). Results will be displayed below. Share your queries by copying and pasting the URL that is generated after each run.
Schema Ready


CREATE TABLE Coffee_Shop (
shop_id INT PRIMARY KEY,
shop_name VARCHAR(50),
city VARCHAR(50),
state CHAR(2));
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 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 Coffee (
coffee_id INT PRIMARY KEY,
Build Schema
Edit Fullscreen
Browser
[;]
Run SQL
Edit Fullscreen
[;]
Query Panel
Use this panel to try to solve the problem with other SQL statements (SELECTs, etc...). Results will be displayed below. Share your queries by copying and pasting the URL that is generated after each run.
Schema Ready

QUESTION 2

A

INSERT INTO Coffee_Shop VALUES
(111,'Brewed Awakening','Beanville','CA'),
(222,'Caffeine Corner','Espresso City','NY'),
(333,'Mugs and Beans','Latteville','MD');

INSERT INTO Supplier VALUES
(10,'Global Beans Inc','USA','Joe Java','joe.java@example.com'),
(20,'EuroRoast','France','Marie Espresso','marie.espresso@example.com'),
(30,'AsiaCoffee Ltd','Japan','Takashi Sato','takashi.sato@example.com');

INSERT INTO Employee VALUES
(101,'John','Smith','2023-01-15','Barista',333),
(202,'Emily','Johnson','2023-03-20','Assistant Manager',222),
(303,'Michael','Brown','2023-05-10','Head Roaster',333);

INSERT INTO Coffee VALUES
(12,333,10,'Midnight Mocha Madness','16.50'),
(13,333,10,'Caramel Latte','13.00'),
(14,111,20,'Hazelnut Hug in a Mug','12.65');

B

```sql
SELECT *
FROM Coffee_Shop;

SELECT *
FROM Supplier;

SELECT *
FROM Employee;

SELECT *
FROM Coffee;
```

Build Schema | Edit Fullscreen | Browser | [;]

Run SQL | Edit Fullscreen | [;]

shop_id	shop_name	city	state
111	Brewed Awakening	Beanville	CA
222	Caffeine Corner	Espresso City	NY
333	Mugs and Beans	Latteville	MD

Record Count: 3; Execution Time: 15ms | View Execution Plan | link

supplier_id	company_name	country	sales_contact_name	email
10	Global Beans Inc	USA	Joe Java	joe.java@example.com
20	EuroRoast	France	Marie Espresso	marie.espresso@example.com
30	AsiaCoffee Ltd	Japan	Takashi Sato	takashi.sato@example.com

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

employee_id	first_name	last_name	hire_date	job_title	shop_id
101	John	Smith	2023-01-15	Barista	333
202	Emily	Johnson	2023-03-20	Assistant Manager	222
303	Michael	Brown	2023-05-10	Head Roaster	333

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

coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
12	333	10	Midnight Mocha Madness	16.5
13	333	10	Caramel Latte	13
14	111	20	Hazelnut Hug in a Mug	12.65

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

QUESTION 3

A

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

3B

```
SELECT *
FROM employee_view;
```

Build Schema | Edit Fullscreen | Browser | [;] | Run SQL | Edit Fullscreen | [;]

employee_id	employee_full_name	hire_date	job_title	shop_id
101	John Smith	2023-01-15	Barista	333
202	Emily Johnson	2023-03-20	Assistant Manager	222
303	Michael Brown	2023-05-10	Head Roaster	333

Record Count: 3; Execution Time: 13ms | View Execution Plan | link

QUESTION 4

A

CREATE INDEX idx_coffee_name ON Coffee (coffee_name);

4B

```
1 SHOW INDEX FROM Coffee;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
coffee	0	PRIMARY	1	coffee_id	A	3	(null)	(null)		BTREE		
coffee	1	shop_id	1	shop_id	A	3	(null)	(null)	YES	BTREE		
coffee	1	supplier_id	1	supplier_id	A	3	(null)	(null)	YES	BTREE		
coffee	1	idx_coffee_name	1	coffee_name	A	3	(null)	(null)	YES	BTREE		

✔ Record Count: 4; Execution Time: 14ms ↱ link

QUESTION 5

A

SELECT shop_name AS Shop, CONCAT(city, ', ',state) AS Location FROM Coffee_Shop
WHERE state IN ('CA','MD');

5B

```
SELECT shop_name AS Shop, CONCAT(city, ', ',state)
AS Location FROM Coffee_Shop
WHERE state IN ('CA','MD');
```

Build Schema | Edit Fullscreen | Browser | [;] Run SQL | Edit Fullscreen | [;]

Shop	Location
Brewed Awakening	Beanville, CA
Mugs and Beans	Latteville, MD

Record Count: 2; Execution Time: 17ms View Execution Plan link

Did this query solve the problem? If so, consider donating $5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!

QUESTION 6

A

SELECT Coffee_Shop.shop_name, Coffee.coffee_name, Coffee.price_per_pound,
Supplier.company_name, Supplier.sales_contact_name, Supplier.email
FROM Coffee
JOIN Coffee_Shop
ON Coffee_Shop.shop_id = Coffee.shop_id JOIN Supplier
ON Supplier.supplier_id = Coffee.supplier_id;

6B

```
SELECT Coffee_Shop.shop_name, Coffee.coffee_name, Coffee.price_per_pound,
Supplier.company_name, Supplier.sales_contact_name, Supplier.email
FROM Coffee
JOIN Coffee_Shop
ON Coffee_Shop.shop_id = Coffee.shop_id JOIN Supplier
ON Supplier.supplier_id = Coffee.supplier_id;
```

Build Schema | Edit Fullscreen | Browser | [;]

Run SQL | Edit Fullscreen | [;]

shop_name	coffee_name	price_per_pound	company_name	sales_contact_name	email
Mugs and Beans	Midnight Mocha Madness	16.5	Global Beans Inc	Joe Java	joe.java@example.com
Mugs and Beans	Caramel Latte	13	Global Beans Inc	Joe Java	joe.java@example.com
Brewed Awakening	Hazelnut Hug in a Mug	12.65	EuroRoast	Marie Espresso	marie.espresso@example.com

Record Count: 3; Execution Time: 6ms | View Execution Plan | link