**A.**



**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 |

**A1:**

**Nora’s Bagel Bin Database Blueprints *(continued)***

**Second Normal Form (2NF)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **BAGEL ORDER** | |  | **BAGEL ORDER LINE ITEM** | |  | **BAGEL** | |
| PK | Bagel Order ID |  | PK / FK | Bagel Order ID |  | PK | Bagel ID |
|  | Order Date | 1:M | PK / FK | Bagel ID | 1:M |  | Bagel Name |
|  | First Name |  |  | Bagel Quantity |  |  | Bagel Description |
|  | Last Name |  |  |  |  |  | Bagel Price |
|  | Address 1 |  |  |  |  |  |  |
|  | Address 2 |  |  |  |  |  |  |
|  | City |  |  |  |  |  |  |
|  | State |  |  |  |  |  |  |
|  | Zip |  |  |  |  |  |  |
|  | Mobile Phone |  |  |  |  |  |  |
|  | Delivery Fee |  |  |  |  |  |  |
|  | Special Notes |  |  |  |  |  |  |

Each column is associated with the primary key in the columns they are logically assigned to and all non-key attributes are functionally dependent on the entire primary key.

The Bagel Order table has a 1 to many relationship with the Bagel Order Line Item table. There can be many Bagel Order Line Items associated with an order but there are can only be 1 order associated with each Bagel Order Line Item.

The Bagel Order Line Item Table has a 1 to many relationship with the Bagel table. There can be many bagels associated with a Bagel Order Line Item but only 1 Bagel Order Line Item for each Bagel.

**A2:**

**Nora’s Bagel Bin Database Blueprints *(continued)***

**Third Normal Form (3NF)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bagel Order** | | |  | **BAGEL ORDER LINE ITEM** | |  | **BAGEL** | |
| PK | Bagel Order ID | |  | PK / FK | Bagel Order ID |  | PK | Bagel ID |
| FK | Customer ID | | 1:M | PK / FK | Bagel ID | 1:M |  | Bagel Name |
|  | Order Date | |  |  | Bagel Quantity |  |  | Bagel Description |
|  | Delivery Fee | |  |  |  |  |  | Bagel Price |
|  | Special Notes | |  |  |  |  |  |  |
|  | 1:M |  |  |  |  |  |  |  |
| **Customer** | | |  |  |  |  |  |  |
| PK | Customer ID | |  |  |  |  |  |  |
|  | First Name | |  |  |  |  |  |  |
|  | Last Name | |  |  |  |  |  |  |
|  | Address 1 | |  |  |  |  |  |  |
|  | Address 2 | |  |  |  |  |  |  |
|  | City | |  |  |  |  |  |  |
|  | State | |  |  |  |  |  |  |
|  | Zip | |  |  |  |  |  |  |
|  | Mobile Phone | |  |  |  |  |  |  |

The Customer data can be repeated in the Bagel Order table and does not depend on the primary key of this table.

I created a new attribute called Customer ID to act as a new primary key, then moved all relevant customer information to the new table Customer.

The Relationships between the Bagel Order, Bagel Order Line Item and Bagel tables stay the same.

The relationship between Customer and Bagel Order is a 1 to many relationship. A Bagel order can only belong to 1 customer; However, a Customer can have multiple Bagel Orders.

**A3:**

**Nora’s Bagel Bin Database Blueprints *(continued)***

**Final Physical Database Model**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bagel Order** | | |  | **BAGEL ORDER LINE ITEM** | | |  | **BAGEL** | |  |
| PK | bagel\_order\_id | INT |  | PK / FK | bagel\_order\_id | INT |  | PK | bagel\_id | CHAR(2) |
| FK | customer\_id | INT | 1:M | PK / FK | bagel\_id | CHAR(2) | 1:M |  | bagel\_name | VARCHAR(30) |
|  | order\_date | DATETIME |  |  | bagel\_quantity | INT |  |  | bagel\_description | VARCHAR(30) |
|  | delivery\_fee | NUMERIC(3,2) |  |  |  |  |  |  | bagel\_price | NUMERIC(2,2) |
|  | special\_notes | VARCHAR(50) |  |  |  |  |  |  |  |  |
|  | 1:M |  |  |  |  |  |  |  |  |  |
| **Customer** | | |  |  |  |  |  |  |  |  |
| PK | customer\_id | INT |  |  |  |  |  |  |  |  |
|  | first\_name | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | last\_name | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | address\_1 | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | address\_2 | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | city | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | state | CHAR(2) |  |  |  |  |  |  |  |  |
|  | zip | NUMERIC(5) |  |  |  |  |  |  |  |  |
|  | mobile\_phone | INT |  |  |  |  |  |  |  |  |

**B1**

Create Table Coffee\_Shop (

shop\_id INT,

shop\_name VARCHAR(50),

city VARCHAR(50),

state CHAR(2),

Primary Key (shop\_id)

);

Create Table Employee (

employee\_id INT,

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,

company\_name VARCHAR(50),

country VARCHAR(30),

sales\_contact\_name VARCHAR(60),

email VARCHAR(50) NOT NULL,

Primary Key (supplier\_id)

);

Create Table Coffee (

coffee\_id INT,

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)

);

INSERTS:

INSERT INTO Coffee\_Shop

VALUES (32, 'Mainstreet', 'Coon Rapids', 'MN'),

(21, 'Downtown', 'St. Paul', 'MN'),

(23, 'Northtown', 'Blaine', 'MN');

INSERT INTO Employee

VALUES (1, 'Jim', 'Cook', '2020-02-01', 'manager', 32),

(2, 'Bob', 'Johnson', '2019-01-01', 'barista', 32),

(3, 'Samantha', 'Reed', '2019-02-02', 'barista', 21);

INSERT INTO Supplier

VALUES(303, 'Carbou', 'USA', 'Ron Douglas', 'ron@test.com'),

(305, 'Steerbucks', 'USA', 'Jenny Black', 'jenny@coffee.com'),

(304, 'Simons', 'USA', 'Jameson Howard', 'james@thiscoffee.com');

INSERT INTO Coffee

VALUES (15, 32, 303, 'Morning Blend', 15.32),

(11, 32, 305, 'French Press', 28.54),

(12, 21, 305, 'Bucking Bean', 55.43);

Screenshots of code below.

A picture containing graphical user interface

Description automatically generated

A picture containing graphical user interface

Description automatically generated

SQL STATEMENTS

**B3**

CONCAT View:

SELECT employee\_id, CONCAT(first\_name, ' ', last\_name) AS employee\_full\_name, hire\_date, job\_title, shop\_id

FROM Employee;

Graphical user interface, text, application

Description automatically generated

**B4**

Create Index:

CREATE INDEX COFFFEE

ON Coffee (coffee\_name);

Graphical user interface, text

Description automatically generated

**B5**

Select From Where:

Select\*

From Employee

Where job\_title = ‘barista’;

Graphical user interface, text, application

Description automatically generated

**B6**

SELECT CS.shop\_name, C.coffee\_name, S.company\_name

From Coffee\_Shop CS

INNER Join Coffee C ON C.shop\_id = CS.shop\_id

INNER Join Supplier S ON S.supplier\_id = C.supplier\_id;

Graphical user interface, application

Description automatically generated