**Nora’s Bagel Bin Database Blueprints**

**Part A:**

**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 |

The Bagel Order table contains atomic and unique values in its first normal form. There is a composite key within the table due to two or more specific entities, particularly the Bagel Order ID and the Bagel ID. This composite key means that the data is functionally dependent. Therefore, more data normalization will stop likely data redundancy.

**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 |  |  |  |  |  |  |

Using the second normal form, the first bagel order now consists of three entities. First, we have the Bagel Order table that contains the bagel order ID, order date, and the customer's information which makes each order unique. The information related to the specifics of the bagel was placed in the Bagel table. Creating another table was necessary since this information will remain the same regardless of the order. The Bagel Order Line Item table divides the initial many-to-many relationship by serving as a bridge between the Bagel Order entity and the Bagel entity. Within the Bagel Order Line Item table is the Bagel Quantity attribute. This attribute that links to the Bagel table, makes sure that every quantity is based on a unique order.

* The relationship between the Bagel Order to Bagel Order Line Item tables is one to many. There are numerous line items in every Bagel Order, but each line item can only have one specific order. This is evident because of the Bagel Order ID key being a primary/foreign key within the Bagel Order Line Item table linking it to the Bagel Order table.
* The relationship between the Bagel Order Line Item table and the Bagel table is one to many. There are numerous Bagel types in every Bagel Order Line Item, but there is only one specific bagel type for every Bagel Order Line Item. This is evident because of the Bagel ID key being a primary/foreign key within the Bagel Order Line Item. Also, the Bagel Quantity being in the Bagel Order Line Item table allows for every quantity of bagels as a line item to be independent of the single specific bagel type that is located in the Bagel table.

**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 | |  |  |  |  |  |  |

This third normal form was made possible by further breaking down the second normal form of the Bagel Order table into two separate tables, the Bagel Order table, and the Customer table. The information related to the specifics of the customer was placed in the Customer table. The Customer table allows the customer’s information to be independent and stops likely data redundancy.

* The relationship between the Bagel Order table and the Customer table is one to many. One customer can place numerous orders, but each order will only have one and just one customer.

**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 | TIMESTAMP |  |  | bagel\_quantity | INT |  |  | bagel\_description | VARCHAR(30) |
|  | delivery\_fee | NUMERIC(4,2) |  |  |  |  |  |  | bagel\_price | NUMERIC(3,2) |
|  | special\_notes | VARCHAR(150) |  |  |  |  |  |  |  |  |
|  | **1:M** |  |  |  |  |  |  |  |  |  |
| **CUSTOMER** | | |  |  |  |  |  |  |  |  |
| PK | customer\_id | INT |  |  |  |  |  |  |  |  |
|  | first\_name | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | last\_name | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | address\_1 | VARCHAR(50) |  |  |  |  |  |  |  |  |
|  | address\_2 | VARCHAR(50) |  |  |  |  |  |  |  |  |
|  | street | VARCHAR(50) |  |  |  |  |  |  |  |  |
|  | city | CHAR(2) |  |  |  |  |  |  |  |  |
|  | zip | CHAR(5) |  |  |  |  |  |  |  |  |
|  | mobile\_phone | CHAR(10) |  |  |  |  |  |  |  |  |

The final physical database model is the previous third normal form with the added data types. Furthermore, the table complies with the SQL standards by modifying all the data types and attributes, ensuring that the data types are given to the relevant attribute.

**Jaunty Coffee Co. Database**

**Part B:**

1. Develop SQL code to create each table as specified in the attached “Jaunty Coffee Co. ERD” by using the code you wrote to create all the tables.

**Graphical user interface, text, application

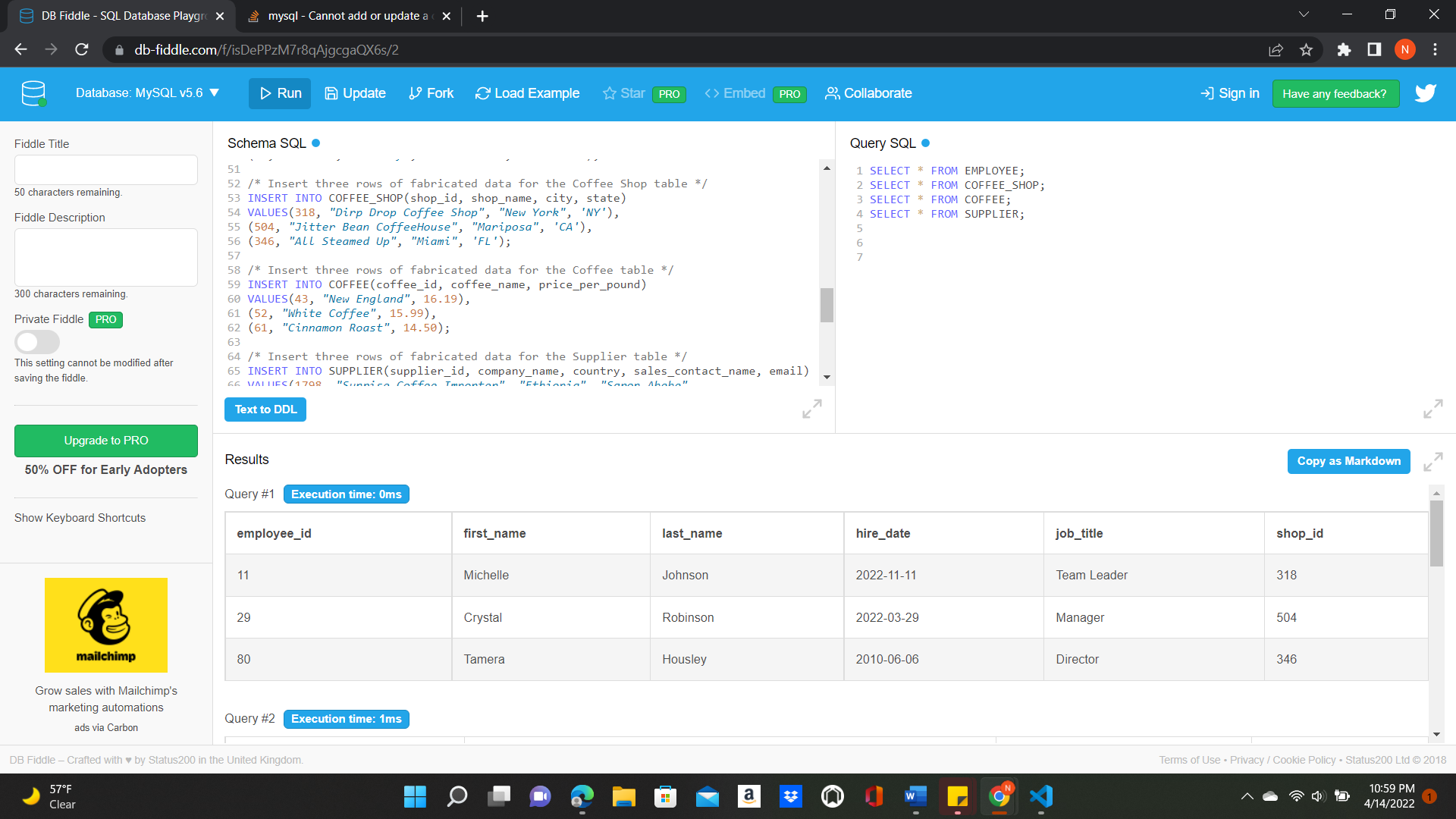
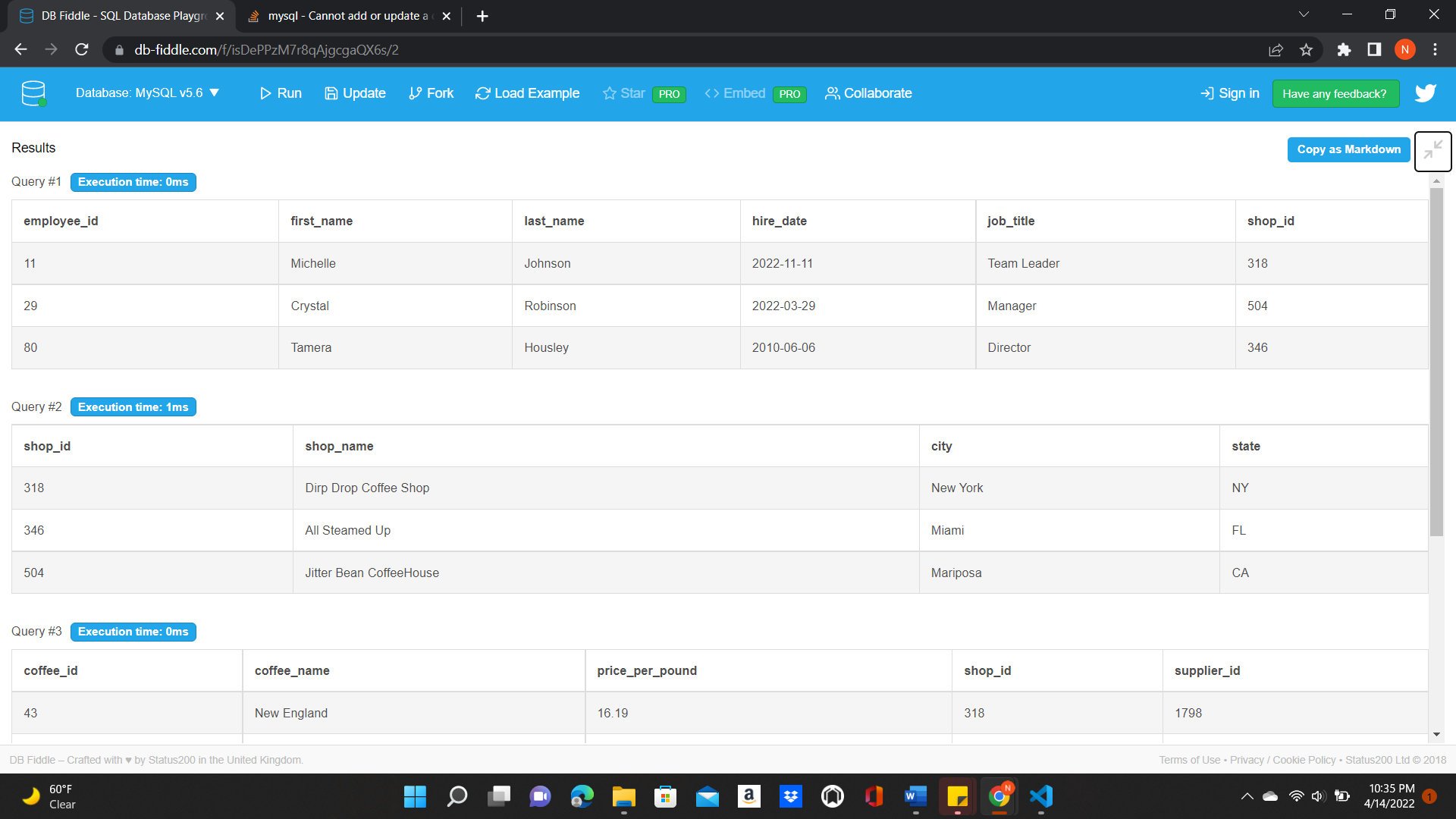
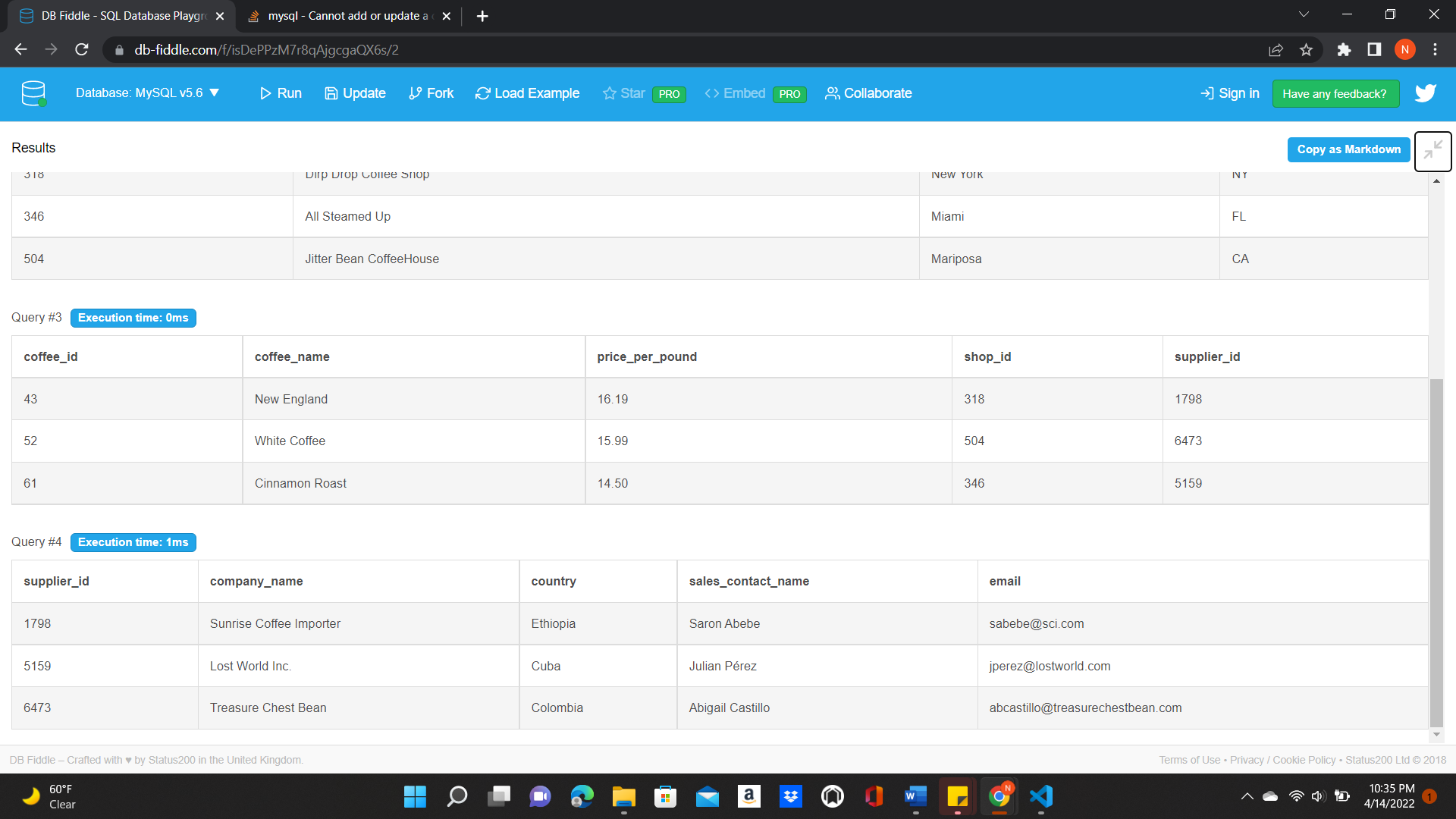
Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **Graphical user interface, text

   Description automatically generated**Develop SQL code to populate each table in the database design document with at least **three** rows of data in each table.



1. Develop SQL code by showing all of the information from the “Employee” table but concatenate each employee’s first and last name, formatted with a space between the first and last name, into a new attribute called employee\_full\_name.

**Graphical user interface, text

Description automatically generated**

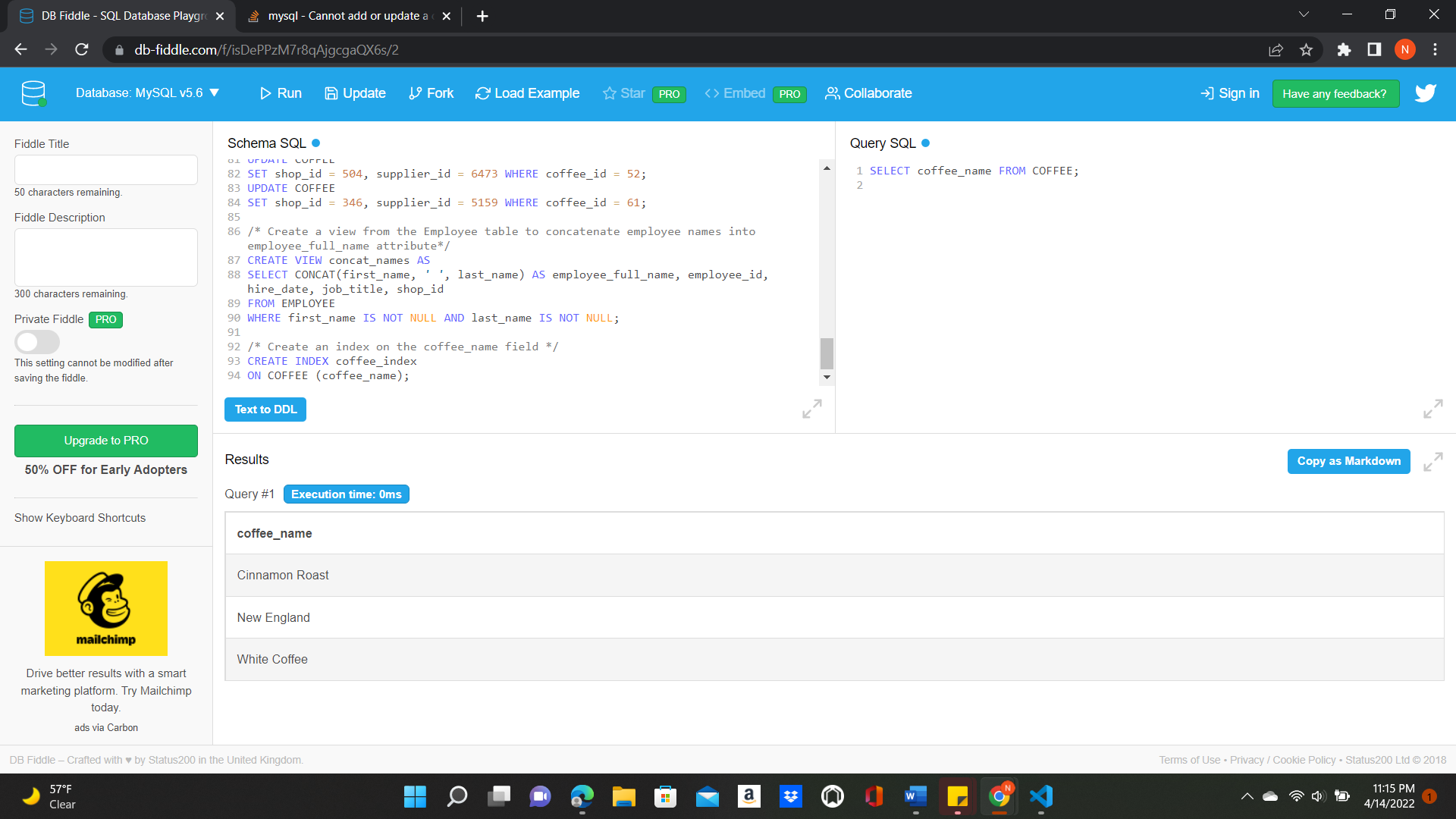
**A screenshot of a computer

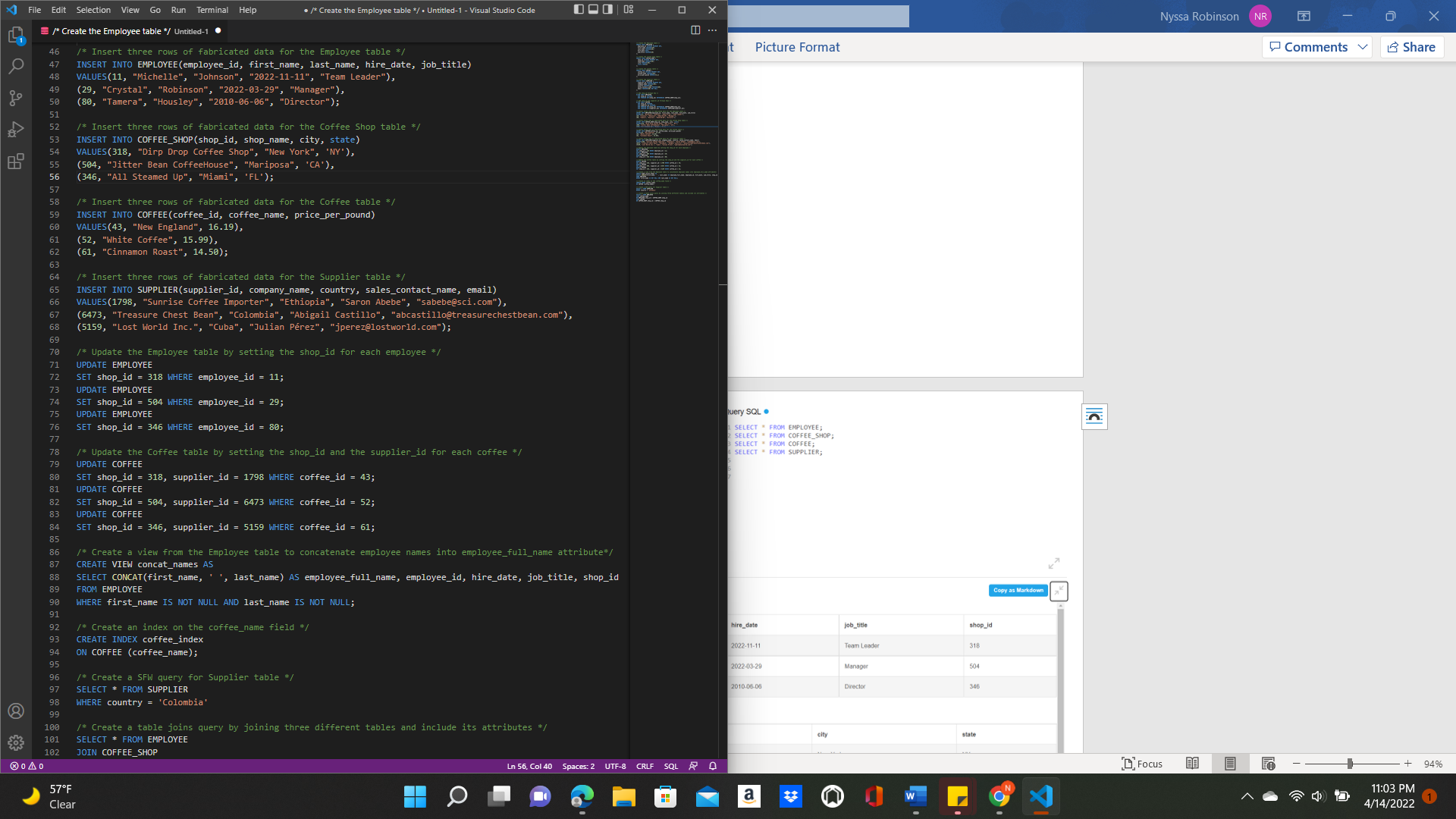
Description automatically generated**

1. Develop SQL code to create an index on the coffee\_name field from the COFFEE table

Graphical user interface, text

Description automatically generated



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

A screenshot of a computer

Description automatically generated

1. **Graphical user interface, text, application

   Description automatically generated**Develop SQL code by joining together **three** different tables and include attributes from all three tables in its output.

**Graphical user interface, text, application

Description automatically generated**

**A screenshot of a computer

Description automatically generated**