**Section A.1:**

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

**A.1.a-b)**

**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 | M:1 |  | 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 |  |  |  |  |  |  |

**A.1.c )**

**Q:** *Explain how you assigned attributes to the 2NF tables and determined the cardinality of the relationships between your 2NF tables*.

**A:** The “Bagel Order” table contains all the information related to the order, based on the information presented in the “Bagel Order Form.” The intersection table contains the foreign key, “Bagel Order ID”, that links to the primary key, “Bagel Order ID”, in the “Bagel Order” table. The cardinality of this relationship is one-to-many; moreover, one order can have many bagel line items. The “Bagel ID” foreign key links to the primary key, “Bagel ID”, in the “Bagel Table.” The cardinality of this relationship is many-to-one; many bagel order line items have one bagel type. Furthermore, this leads to needing a “Bagel Quantity” attribute in the “Bagel Order Line Item” table to allow us to quantify the bagel types per order. Lastly, the “Bagel” table contains all information related to the bagel, minus the bagel quantity.

**Section A.2:**

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

**A.2.a-d )**

**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 | M:1 |  | Bagel Name |
|  | Order Date | |  |  | Bagel Quantity |  |  | Bagel Description |
|  | Delivery Fee | |  |  |  |  |  | Bagel Price |
|  | Special Notes | |  |  |  |  |  |  |
|  | M:1 |  |  |  |  |  |  |  |
| **CUSTOMER** | | |  |  |  |  |  |  |
| PK | Customer ID | |  |  |  |  |  |  |
|  | First Name | |  |  |  |  |  |  |
|  | Last Name | |  |  |  |  |  |  |
|  | Address 1 | |  |  |  |  |  |  |
|  | Address 2 | |  |  |  |  |  |  |
|  | City | |  |  |  |  |  |  |
|  | State | |  |  |  |  |  |  |
|  | Zip | |  |  |  |  |  |  |
|  | Mobile Phone | |  |  |  |  |  |  |

**A.2.e )**

**Q:** *Explain how you assigned attributes to the 3NF tables and determined the cardinality of the relationships between your 3NF tables.*

**A:** The “Bagel Order” table, “Bagel Order Line Item” table, and “Bagel” table have the exact same relationships as 2NF. The main difference is that the customer attributes were extracted from the “Bagel Order” table and a new “Customer” table was created. This created a new many-to-one relationship between the “Bagel Order” table and the “Customer” table; one customer can have many orders.

**Section A.3:**

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

**A.3.a-b )**

**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) | M:1 |  | bagel\_name | VARCHAR(30) |
|  | order\_date | TIMESTAMP |  |  | bagel\_quantity | INT |  |  | bagel\_description | VARCHAR(30) |
|  | delivery\_fee | NUMERIC |  |  |  |  |  |  | bagel\_price | NUMERIC |
|  | special\_notes | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | M:1 |  |  |  |  |  |  |  |  |  |
| **CUSTOMER** | | |  |  |  |  |  |  |  |  |
| PK | customer\_id | INT |  |  |  |  |  |  |  |  |
|  | first\_name | VARCHAR(15) |  |  |  |  |  |  |  |  |
|  | last\_name | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | address\_1 | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | address\_2 | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | city | VARCHAR(30) |  |  |  |  |  |  |  |  |
|  | state | CHAR(2) |  |  |  |  |  |  |  |  |
|  | zip | VARCHAR(10) |  |  |  |  |  |  |  |  |
|  | mobile\_phone | VARCHAR(10) |  |  |  |  |  |  |  |  |

**Section B:**

**B.1)** Develop SQL code to create each table as specified in the attached “Jaunty Coffee Co. ERD” by doing the following:

a. Provide the SQL code you wrote to create all the tables.

b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server’s response

**CODE:**

**CREATE** **DATABASE** **IF** **NOT** **EXISTS** jaunty\_coffee\_co**;**

**USE** jaunty\_coffee\_co**;**

**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**)**

**);**

**RESULTS:**

Graphical user interface, application

Description automatically generated

**B.2)** Develop SQL code to populate each table in the database design document by doing the following:

a. Provide the SQL code you wrote to populate the tables with at least three rows of data in each table.

b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server’s response.

**CODE: (COFFEE SHOP TABLE)**

**INSERT** **INTO** COFFEE\_SHOP

**VALUES** **(**1**,** 'Starbucks'**,** 'Pittsburgh'**,** 'PA'**);**

**INSERT** **INTO** COFFEE\_SHOP

**VALUES** **(**2**,** 'Cupka Joes'**,** 'Pittsburgh'**,** 'PA'**);**

**INSERT** **INTO** COFFEE\_SHOP

**VALUES** **(**3**,** 'Some Coffee Shop'**,** 'Pittsburgh'**,** 'PA'**);**

**RESULTS: (COFFEE SHOP TABLE)**

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**CODE: (EMPLOYEE TABLE)**

**INSERT** **INTO** EMPLOYEE

**VALUES** **(**1**,** 'Eddie'**,** 'Blanciak'**,** '2022-06-19'**,** 'Developer'**,** 1**);**

**INSERT** **INTO** EMPLOYEE

**VALUES** **(**2**,** 'Joe'**,** 'Smith'**,** '2022-06-19'**,** 'Java Developer'**,** 2**);**

**INSERT** **INTO** EMPLOYEE

**VALUES** **(**3**,** 'Mary'**,** 'Toy'**,** '2022-06-19'**,** 'C++ Developer'**,** 3**);**

**RESULTS: (EMPLOYEE TABLE)**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, table

Description automatically generated

**CODE: (SUPPLIER TABLE)**

**INSERT** **INTO** SUPPLIER

**VALUES** **(**1**,** 'Big Coffee Supplier'**,** 'USA'**,** 'Salesman Joe'**,** 'joe@bigcoffeesupplier.com'**);**

**INSERT** **INTO** SUPPLIER

**VALUES** **(**2**,** 'Medium Coffee Supplier'**,** 'USA'**,** 'Salesman Ed'**,** 'ed@mediumcoffeesupplier.com'**);**

**INSERT** **INTO** SUPPLIER

**VALUES** **(**3**,** 'Small Coffee Supplier'**,** 'USA'**,** 'Salesman Tim'**,** 'tim@smallcoffeesupplier.com'**);**

**RESULTS: (SUPPLIER TABLE)**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, table

Description automatically generated

**CODE: (COFFEE TABLE)**

**INSERT** **INTO** COFFEE

**VALUES** **(**1**,** 1**,** 1**,** 'Really Good Coffee'**,** 10.50**);**

**INSERT** **INTO** COFFEE

**VALUES** **(**2**,** 2**,** 2**,** 'Really Okay Coffee'**,** 5.50**);**

**INSERT** **INTO** COFFEE

**VALUES** **(**3**,** 3**,** 2**,** 'Really Bad Coffee'**,** 1.50**);**

**RESULTS: (COFFEE TABLE)**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, table

Description automatically generated

**B.3)** Develop SQL code to create a view by doing the following:

a. Provide the SQL code you wrote to create your view. The view should show 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.

b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server’s response.

**CODE:**

**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**;**

**RESULTS:**

Graphical user interface, application

Description automatically generated

Graphical user interface, application, table

Description automatically generated

**B.4)** Develop SQL code to create an index on the coffee\_name field by doing the following:

a. Provide the SQL code you wrote to create your index on the coffee\_name field from the “Coffee” table.

b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server’s response.

**CODE:**

**CREATE** **INDEX** coffee\_name

**ON** COFFEE **(**coffee\_name**);**

**RESULTS:**

Graphical user interface

Description automatically generated with low confidence

Table

Description automatically generated

**B.5)** Develop SQL code to create an SFW (SELECT–FROM–WHERE) query for any of your tables or views by doing the following:

a. Provide the SQL code you wrote to create your SFW query.

b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server’s response.

**CODE:**

**SELECT** first\_name **AS** 'First Name'**,** last\_name **AS** 'Last Name'**,** job\_title **AS** 'Job Title'

**FROM** EMPLOYEE

**WHERE** job\_title **LIKE** '%developer'**;**

**RESULTS:**

Background pattern

Description automatically generated

Table

Description automatically generated

**B.6)** Develop SQL code to create a query by doing the following:

a. Provide the SQL code you wrote to create your table joins query. The query should join together three different tables and include attributes from all three tables in its output.

b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server’s response.

**CODE:**

**SELECT** COFFEE\_SHOP**.**shop\_name **AS** 'Shop Name'**,** COFFEE**.**coffee\_name **AS** 'Coffee Brand Name'**,** COFFEE**.**price\_per\_pound **AS** 'Price Per LB'**,** SUPPLIER**.**company\_name **AS** 'Coffee Supplier'

**FROM** **((**COFFEE

**INNER** **JOIN** COFFEE\_SHOP **ON** COFFEE**.**shop\_id **=** COFFEE\_SHOP**.**shop\_id**)**

**INNER** **JOIN** SUPPLIER **ON** COFFEE**.**supplier\_id **=** SUPPLIER**.**supplier\_id**);**

**RESULTS:**

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated