Data Management – Applications – C170

By

Brandon Jones

This document contains my submission for the C170 performative assessment. I have included the instructions, step by step and have included my solutions. Images have been attached where applicable, otherwise my text solution has been denoted by red text color.

Scenario

You are a database designer and developer who has been hired by two local businesses, Nora's Bagel Bin and Jaunty Coffee Co., to build databases to help them manage their businesses. First, you will design a normalized physical database model to store data for Nora's Bagel Bin's ordering system. Then, you will use an existing database design document for Jaunty Coffee Co. to create its database. Once the tables have been built, you will load them with sample data and create a view and an index to protect and improve query performance. Finally, you will create both a simple query and a more complex table joins query to produce meaningful reports from the newly created database.

Nora's Bagel Bin Normalized Physical Database Model

A. Construct a normalized physical database model to represent the ordering process for Nora's Bagel Bin by doing the following:

1. Complete the second normal form (2NF) section of the attached "Nora's Bagel Bin Database Blueprints" document by doing the following:

Assign each attribute from the 1NF table into the correct 2NF table and describe the relationship between the two pairs of 2NF tables by indicating their cardinality in each of the dotted cells: one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), or many-to-many (M:M).

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

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

1:M

BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Bagel Quantity

M:1

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

I assigned each attribute by starting with adding every attribute that only applied to bagel to the bagel 2NF table. Since we have a line item 2NF table, I knew that our Bagel Quantity would belong to this table instead of the Bagel Order table. The last step then, was adding each attribute that applied to the bagel order to the Bagel Order 2NF table. A bagel order can have multiple line items and one bagel can exist in multiple line items, however only one bagel can exist in each line item. For this reason, the relationship between bagel order and line item is a one to many, and the relationship between a line item and a bagel is a many to one.

2. Complete the third normal form (3NF) section of the attached "Nora's Bagel Bin Database Blueprints" document by doing the following:

a. Assign each attribute from your 2NF "Bagel Order" table into one of the new 3NF tables. Copy all other information from your 2NF diagram into the 3NF diagram.

b. Provide each 3NF table with a name that reflects its contents.

c. Create a new field that will be used as a key linking the two 3NF tables you named in part A2b. Ensure that your primary key (PK) and foreign key (FK) fields are in the correct locations in the 3NF diagram.

d. Describe the relationships between the 3NF tables by indicating their cardinality in each of the dotted cells: one-to-one (1:1), one-to-many (1:M), many-to-one (M:1), or many-to-many (M:M).

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

Third Normal Form (3NF)

BAGEL ORDER	
PK	Bagel Order ID
FK	Customer ID
	Order Date
	Delivery Fee
	Special Notes

1:M

BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Bagel Quantity

M:1

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

M:1

CUSTOMER	
PK	Customer ID
	First Name
	Last Name
	Address 1
	Address 2
	City
	State
	Zip
	Mobile Phone

Since in the 2NF mockup, our bagel order table had repeating data that did not pertain to the bagel order, I took all of the repeating customer data and placed that into a new table, named Customer and added a Customer ID that acted as a primary key for Customer and a foreign key for Bagel Order. The cardinality of the pre-existing tables is the same as the last example, however we do have a new relationship between Bagel Order and customer. This relationship is a many to one, as a one customer can have many bagel orders, however each bagel order must have only one customer.

3. Complete the "Final Physical Database Model" section of the attached “Nora’s Bagel Bin Database Blueprints” document by doing the following:

a. Copy the table names and cardinality information from your 3NF diagram into the “Final Physical Database Model” and rename the attributes.

b. Assign one of the following five data types to each attribute in your 3NF tables: CHAR(), VARCHAR(), TIMESTAMP, INTEGER, or NUMERIC(). Each data type must be used at least once.

Final Physical Database Model

BAGEL ORDER		
PK	bagel_order_id	INT
FK	customer_id	INT
	order_date	TIMESTAMP
	delivery_fee	INT
	special_notes	VARCHAR(50)

1:M

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

M:1

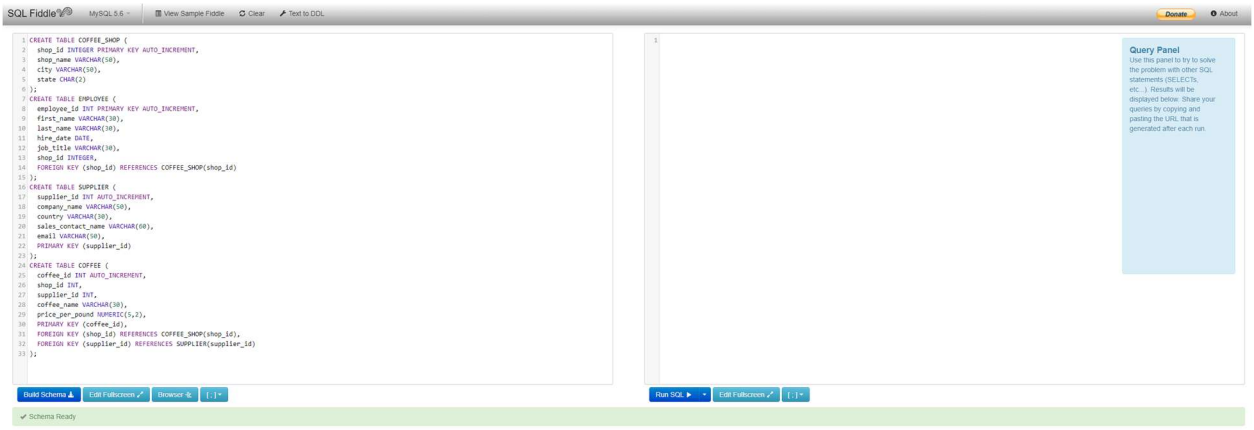
BAGEL		
PK	bagel_id	CHAR(2)
	bagel_name	VARCHAR(50)
	bagel_description	VARCHAR(50)
	bagel_price	NUMERIC(5,2)

M:1

CUSTOMER		
PK	customer_id	INT
	first_name	VARCHAR(50)
	last_name	VARCHAR(50)
	address_1	VARCHAR(50)
	address_2	VARCHAR(50)
	city	VARCHAR(50)
	state	VARCHAR(50)
	zip	VARCHAR(50)
	mobile_phone	VARCHAR(50)

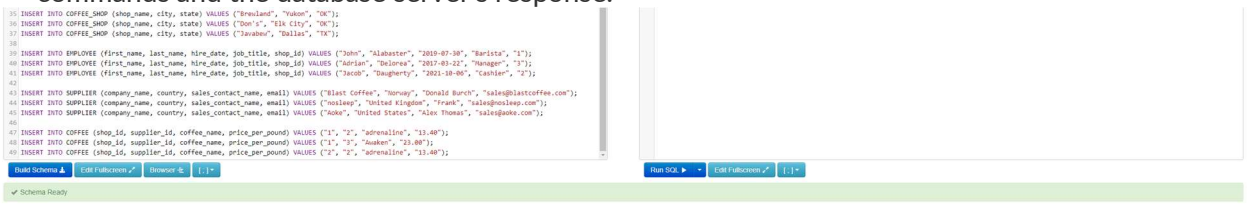
B. Create a database using the attached "Jaunty Coffee Co. ERD" by doing the following:

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.

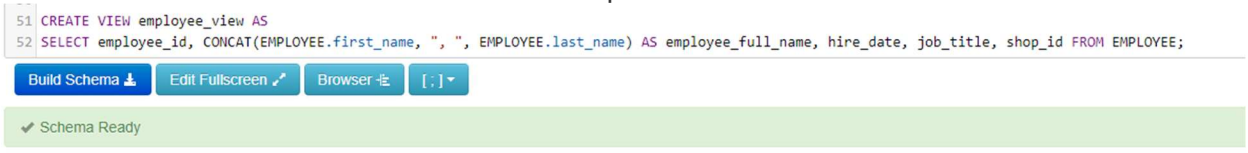


2. Develop SQL code to populate *each* table in the database design document by doing the following:

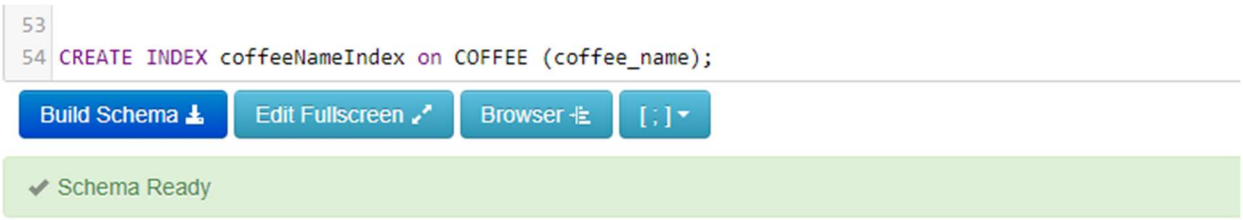
   a. Provide the SQL code you wrote to populate the tables with *at least* **three** rows of data in *each* table.
   b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server's response.



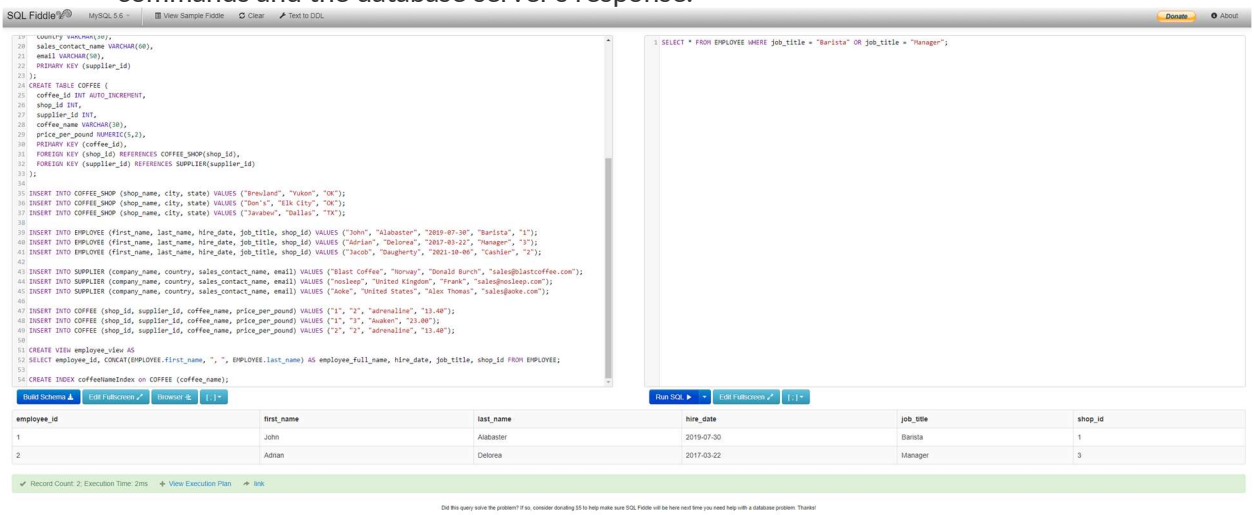
3. Develop SQL code to create a view by doing the following:
   a. Provide the SQL code you wrote to create your view. The view should show *all* of the information from the "Employee" table but concatenate *each* employee's first and last name, formatted with a space between the first and last name, 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.



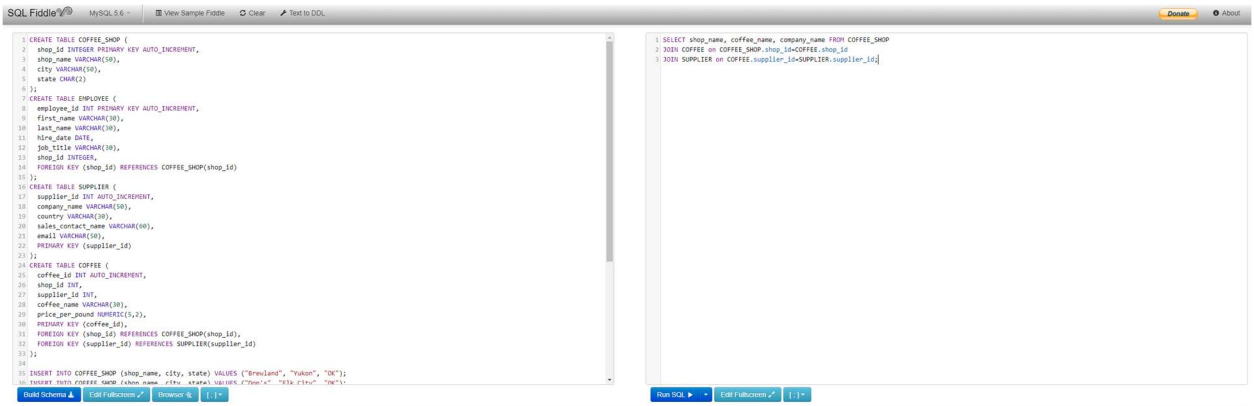
4. Develop SQL code to create an index on the coffee_name field by doing the following:
   a. Provide the SQL code you wrote to create your index on the coffee_name field from the "Coffee" table.
   b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server's response.



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.



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.



shop_name	coffee_name	company_name
Brewland	adrenaline	nosleep
Brewland	Awaken	Aoke
Don's	adrenaline	nosleep

Record Count: 3, Execution Time: 6ms View Execution Plan link