TASK 1: NORMALIZATION AND DATABASE DESIGN
C170 – Performance Assessment

James Blankenship
#003462522

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

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:
a. Assign each attribute from the 1NF table into the correct 2NF table.
b. 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).
c. Explain how you assigned attributes to the 2NF tables and determined the cardinality of the relationships between your 2NF tables.

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

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
	Bagel Quantity
	Special Notes

1:M

BAGEL ORDER LINE ITEM	
PK / FK	Bagel Order ID
PK / FK	Bagel ID
	Delivery Fee

M:1

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

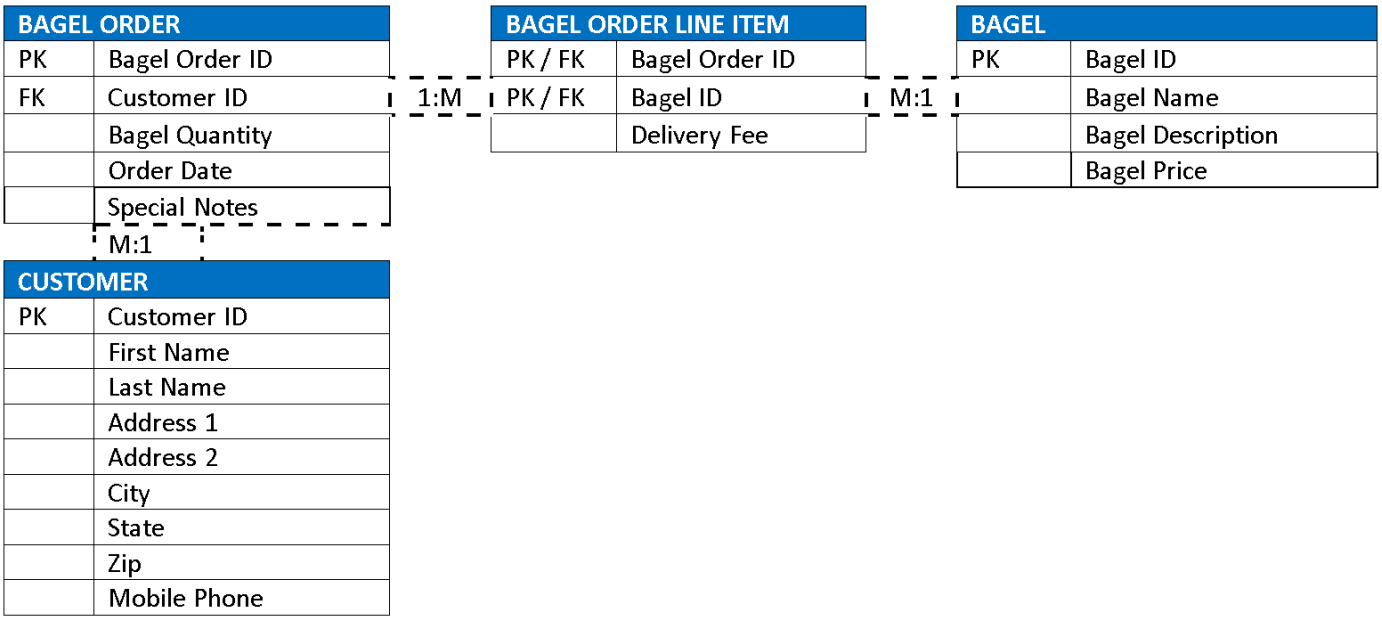
I assigned the attributes to the corresponding tables so none of the non-key attributes are functionally dependent on any candidate key. The Bagel Order table attributes were placed there because they relate to the table and can be dependent on the primary key. The Bagel table's attributes are also dependent on the primary key, which can tell the attributes value from the primary key "Bagel ID". Bagel Order Line Items table's attribute "Delivery Fee" depends on the other attributes "Bagel Order ID" and "Bagel ID". The cardinality between the Bagel Order table and the Bagel Order Line Item is a 1:M relationship because one bagel order can have multiple bagel order line items. The cardinality between the Bagel Order Line Item table and the Bagel table is a M:1 relationship because one bagel can have multiple bagel order line items.

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.

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

Third Normal Form (3NF)



I assigned some of the attributes in the Bagel Order to a new table named "Customer" because all the attributes are dependent on a Customer ID. The cardinality stays the same for the previous tables, and the new cardinality between the Bagel Order table and the Customer table is a M:1 because many bagel orders may be placed for each 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.

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

Final Physical Database Model

BAGEL ORDER		
PK	bagel_order_id	INTEGER
FK	customer_id	INTEGER
	bagel_quantity	INTEGER
	order_date	TIMESTAMP
	special_notes	VARCHAR(100)

1:M

BAGEL ORDER LINE ITEM		
PK / FK	bagel_order_id	INT
PK / FK	bagel_id	CHAR(2)
	delivery_fee	NUMERIC(4,2)

M:1

BAGEL		
PK	bagel_id	CHAR(2)
	bagel_name	VARCHAR(25)
	bagel_description	VARCHAR(100)
	bagel_price	NUMERIC(4,2)

M:1

CUSTOMER		
PK	customer_id	INTEGER
	first_name	VARCHAR(20)
	last_name	VARCHAR(30)
	address_1	VARCHAR(50)
	address_2	VARCHAR(50)
	city	VARCHAR(15)
	state	CHAR(2)
	zip	CHAR(5)
	mobile_phone	CHAR(10)

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:

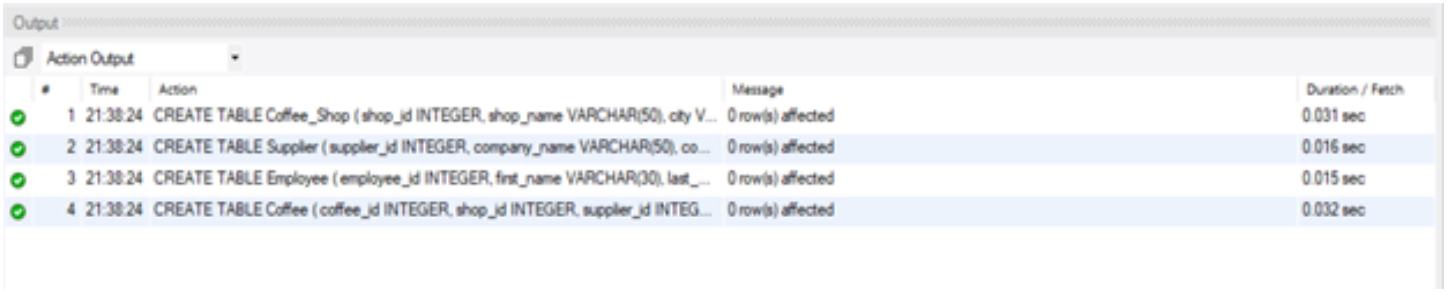
a. Provide the SQL code you wrote to create all the tables.

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

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

CREATE TABLE Supplier (
supplier_id INTEGER,
company_name VARCHAR(50),
country VARCHAR(30),
sales_contact_name VARCHAR(60),
email VARCHAR(50) NOT NULL,
PRIMARY KEY(supplier_id)
);
```

```
17
18 • CREATE TABLE Employee (
19     employee_id INTEGER,
20     first_name VARCHAR(30),
21     last_name VARCHAR(30),
22     hire_date DATE,
23     job_title VARCHAR(30),
24     shop_id INTEGER,
25     PRIMARY KEY(employee_id),
26     FOREIGN KEY(shop_id) REFERENCES Coffee_Shop(shop_id)
27     );
28
```

```
28
29 • CREATE TABLE Coffee (
30     coffee_id INTEGER,
31     shop_id INTEGER,
32     supplier_id INTEGER,
33     coffee_name VARCHAR(30),
34     price_per_pound NUMERIC(5,2),
35     PRIMARY KEY(coffee_id),
36     FOREIGN KEY(shop_id) REFERENCES Coffee_Shop(shop_id),
37     FOREIGN KEY(supplier_id) REFERENCES Supplier(supplier_id)
38     );
39
```

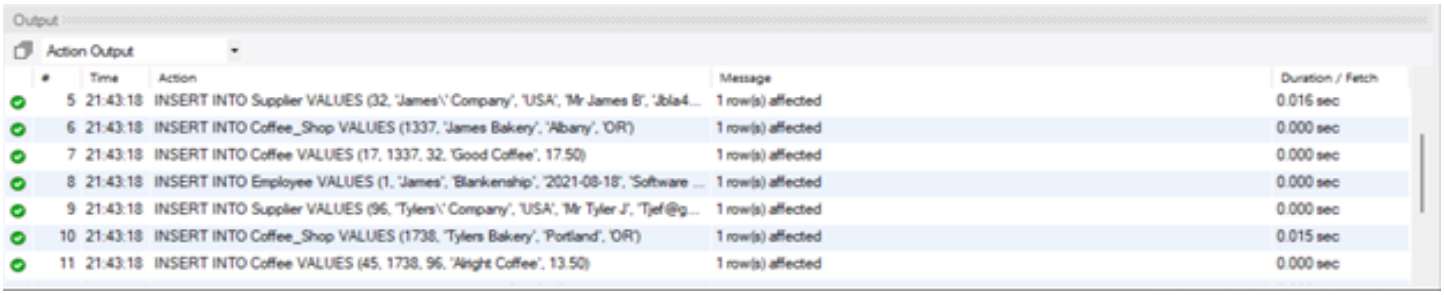


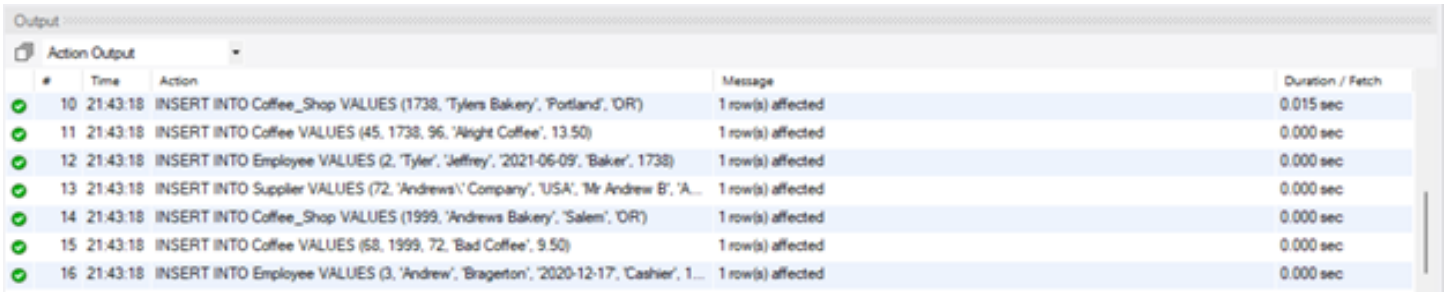
2. Develop SQL code to populate each table in the database design document by doing the following:

Note: This data is not provided. You will be fabricating the data for this step.

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.

```sql
-- First rows in each table

INSERT INTO Supplier
VALUES (32, 'James\' Company', 'USA', 'Mr James B', 'Jbla484@wgu.edu');

INSERT INTO Coffee_Shop
VALUES (1337, 'James Bakery', 'Albany', 'OR');

INSERT INTO Coffee
VALUES (17, 1337, 32, 'Good Coffee', 17.50);

INSERT INTO Employee
VALUES (1, 'James', 'Blankenship', '2021-08-18', 'Software Engineer', 1337);



-- Second rows in each table

INSERT INTO Supplier
VALUES (96, 'Tylers\' Company', 'USA', 'Mr Tyler J', 'Tjef@google.com');

INSERT INTO Coffee_Shop
VALUES (1738, 'Tylers Bakery', 'Portland', 'OR');

INSERT INTO Coffee
VALUES (45, 1738, 96, 'Alright Coffee', 13.50);

INSERT INTO Employee
VALUES (2, 'Tyler', 'Jeffrey', '2021-06-09', 'Baker', 1738);



-- Third rows in each table

INSERT INTO Supplier
VALUES (72, 'Andrews\' Company', 'USA', 'Mr Andrew B', 'Abra@google.com');

INSERT INTO Coffee_Shop
VALUES (1999, 'Andrews Bakery', 'Salem', 'OR');

INSERT INTO Coffee
VALUES (68, 1999, 72, 'Bad Coffee', 9.50);

INSERT INTO Employee
VALUES (3, 'Andrew', 'Bragerton', '2020-12-17', 'Cashier', 1999);

```



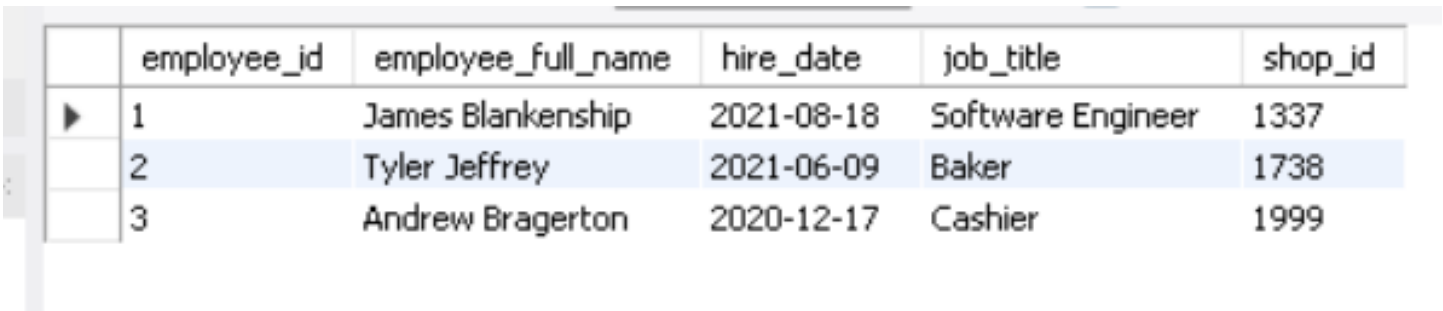


3. Develop SQL code to create a view by doing the following:

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

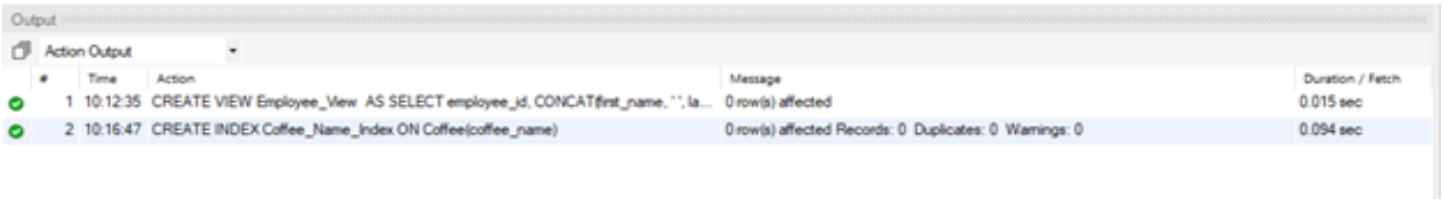
```
1  CREATE VIEW Employee_View
2  AS SELECT employee_id, CONCAT(first_name, ' ', last_name) AS employee_full_name,
3  hire_date,
4  job_title,
5  shop_id
6  FROM Employee;
```



	employee_id	employee_full_name	hire_date	job_title	shop_id
▶	1	James Blankenship	2021-08-18	Software Engineer	1337
	2	Tyler Jeffrey	2021-06-09	Baker	1738
	3	Andrew Bragerton	2020-12-17	Cashier	1999

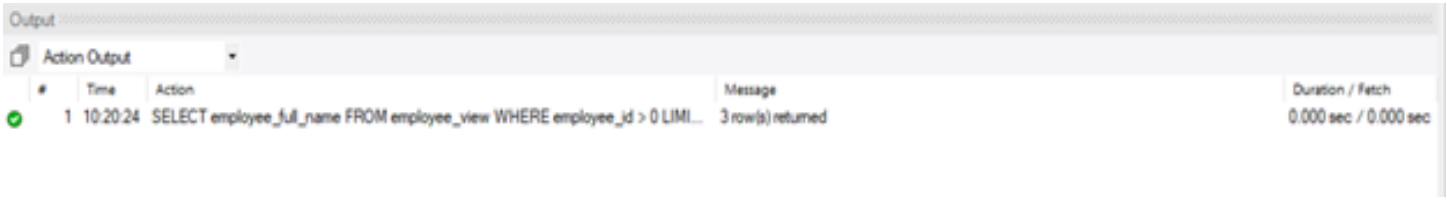
4. Develop SQL code to create an index on the coffee_name field by doing the following:
a. Provide the SQL code you wrote to create your index on the coffee_name field from the “Coffee” table.
b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database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

```
1 •  CREATE INDEX Coffee_Name_Index
2    ON Coffee(coffee_name);
```



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.

```
1 •  SELECT employee_full_name
2    FROM employee_view
3    WHERE employee_id > 0;
```



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.

```sql
SELECT
    E.employee_id,
    E.employee_full_name,
    E.hire_date,
    E.job_title,
    E.shop_id,
    CS.shop_name,
    CS.city,
    CS.state,
    C.coffee_id,
    C.supplier_id,
    C.coffee_name,
    C.price_per_pound
FROM Employee_View AS E
INNER JOIN Coffee_Shop AS CS ON E.shop_id = CS.shop_id
INNER JOIN Coffee AS C ON CS.shop_id = C.shop_id;
```