Shakira Johnson
3/14/2023
WGU C170

VHT2: Normalization and Database Design

A. Normalized physical database model to represent the ordering process for Nora's Bagel Bin

1a, 1b.

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
	Quantity

M:1

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

1c.

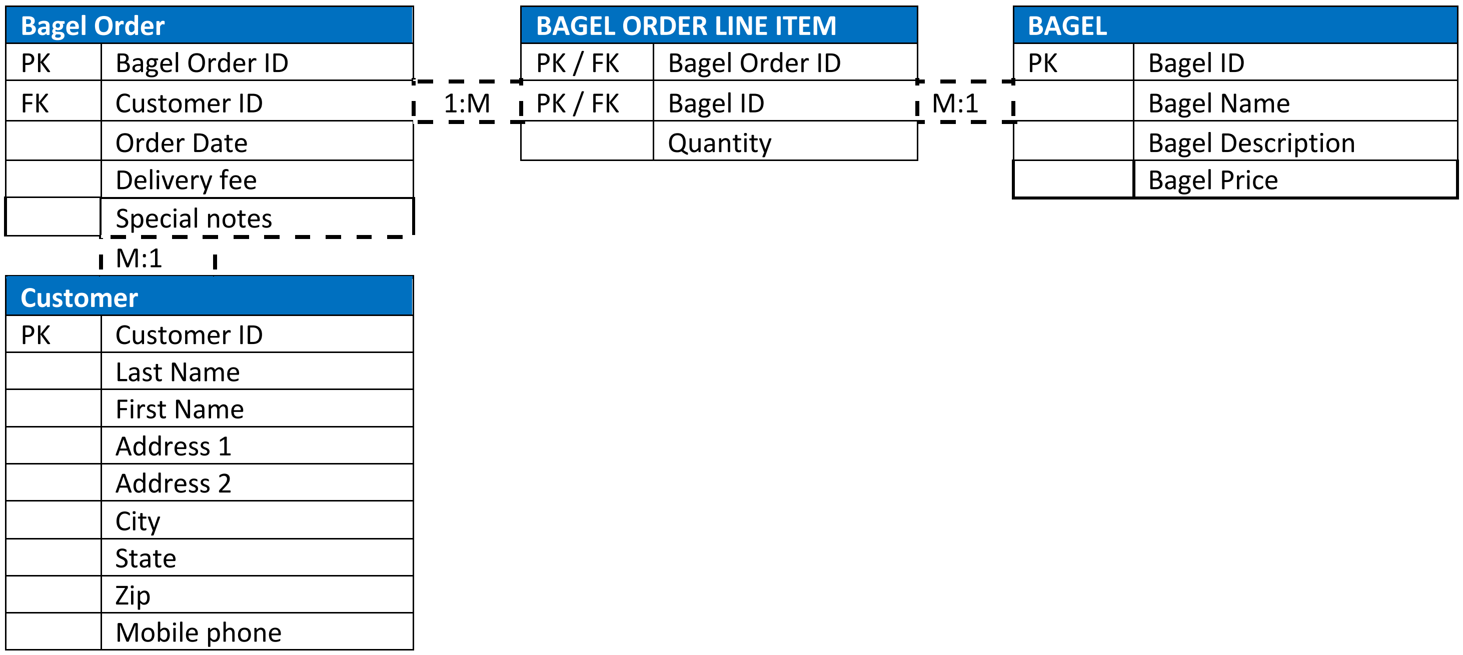
I assigned attributes to the second normal form relation by looking at the attributes of the first normal form table and seeing how they were related to the composite primary key of Bagel ID and bagel order ID in the 1NF table. My goal was to make sure that the attributes are functionally dependent on the composite primary key and not just part of the composite primary key according to the definition of second normal form. I noticed that the attributes bagel name, description, and price in the 1NF table could be determined by the Bagel ID alone instead of using both primary keys. I then determined that these attributes could be put into a separate table using Bagel ID as the primary key. Another table was also added because there were attributes in the 1NF table that could be determined with only the bagel Order ID. The table is called "Bagel Order" and uses Bagel Order ID as the primary key. The orders quantity could only be determined by using both the bagel Order Id and the Bagel Id as a composite primary key in the Bagel Order Line Item table.

The cardinality was determined by looking at the relationship between the relations. A bagel order can contain many line items within one order, so I determined that the cardinality between Bagel Order and Bagel Order Line Item is 1 to many. There is a many to one relationship between Bagel Order Line Item and Bagel because many bagel order line items can have 1 bagel type and 1 bagel may be in many different order line items.

Nora’s Bagel Bin Database Blueprints

2a, 2b, 2c, 2d.

Third Normal Form (3NF)



Bagel Order	
PK	Bagel Order ID
FK	Customer ID
	Order Date
	Delivery fee
	Special notes

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

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

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

2e.

While observing the second normal form table I noticed that there were two separate entities being represented in the bagel order table. The second normal form Bagel Order table includes order information and customer information. According to Dr. Daniel Soper’s YouTube video ‘The Relational Model’, it is a general rule that a well-formed table will not encompass more than one business concept. I decided that it was best to split the attributes related to bagel orders and the attributes related to customers into separate tables. To achieve third normal form the table ‘Customer’ was added. The Customer table also prevents transitive dependencies where an attribute can be determined not only by the primary key but also by another non key attribute. I also wanted to prevent repeating data because if a customer were to place multiple orders in the second normal form Bagel Order table, then their name, address, and phone would have to be repeated within the table. An attribute called Customer ID was added to the Customer table as a primary key for uniqueness and to make each specific customer easily identifiable.

Because there is another table added there must be another cardinality. I determined that the cardinality between Bagel Order and Customer is Many to One because 1 customer can have many bagel orders or in other words many bagel orders can only have one customer per order.

Nora’s Bagel Bin Database Blueprints

3a, 3b.

Final Physical Database Model with Data Types

Bagel Order		
PK	bagel_order_id	INT
FK	bagel_id	CHAR(2)
	order_date	TIMESTAMP
	delivery_fee	INT
	special_notes	VARCHAR(255)

1:M

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

M:1

BAGEL		
PK	bagel_id	CHAR(2)
	bagel_name	VARCHAR(30)
	bagel_description	VARCHAR(30)
	bagel_price	NUMERIC(3,2)

M:1

Customer		
PK	customer_id	INT
	last_name	VARCHAR(50)
	first_name	VARCHAR(50)
	address_1	VARCHAR(50)
	address_2	VARCHAR(50)
	city	VARCHAR(30)
	state	CHAR(2)
	zip	VARCHAR(10)
	mobile_phone	VARCHAR(10)

B. Create A database

1a. SQL code to create each table as specified in “Jaunty Coffee Co. ERD”.

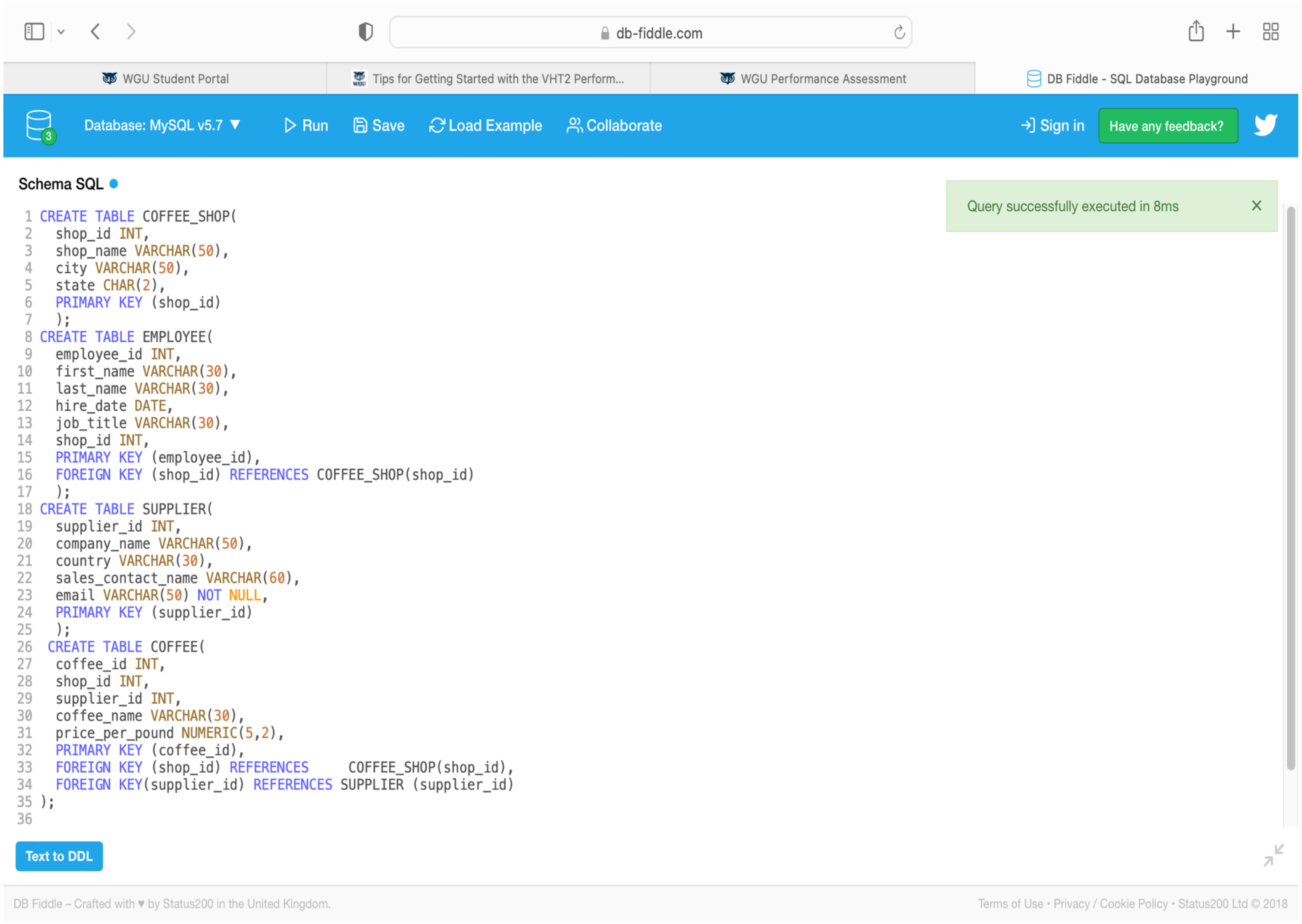
```
CREATE TABLE COFFEE_SHOP(
  shop_id INT,
  shop_name VARCHAR(50),
  city VARCHAR(50),
  state CHAR(2),
  PRIMARY KEY (shop_id)
  );
```

```
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)
);
```

1b. Screenshot of SQL commands and database server response for creating tables.



Query successfully executed in 8ms

```sql
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)
);
```

DB Fiddle – Crafted with ♥ by Status200 in the United Kingdom.

Terms of Use • Privacy / Cookie Policy • Status200 Ltd © 2018

2a. SQL code to populate tables with at least three rows of data.

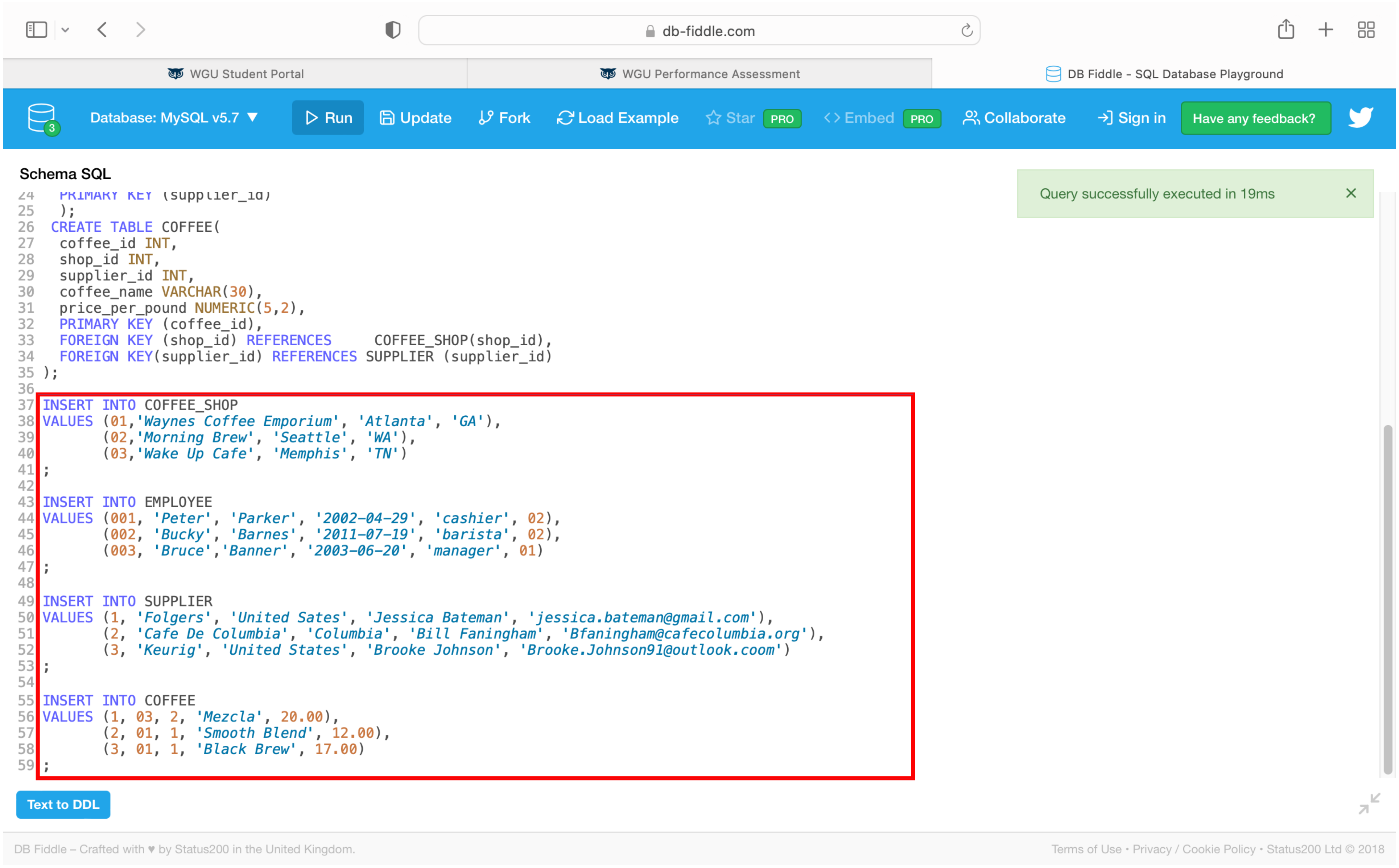
```
INSERT INTO COFFEE_SHOP
VALUES (01,'Waynes Coffee Emporium', 'Atlanta', 'GA'),
        (02,'Morning Brew', 'Seattle', 'WA'),
        (03,'Wake Up Cafe', 'Memphis', 'TN')
;
```

```
INSERT INTO EMPLOYEE
VALUES (001, 'Peter', 'Parker', '2002-04-29', 'cashier', 02),
        (002, 'Bucky', 'Barnes', '2011-07-19', 'barista', 02),
        (003, 'Bruce','Banner', '2003-06-20', 'manager', 01)
;
```

```
INSERT INTO SUPPLIER
VALUES (1, 'Folgers', 'United Sates', 'Jessica Bateman', 'jessica.bateman@gmail.com'),
        (2, 'Cafe De Columbia', 'Columbia', 'Bill Faningham', 'Bfaningham@cafecolumbia.org'),
        (3, 'Keurig', 'United States', 'Brooke Johnson', 'Brooke.Johnson91@outlook.coom')
;
```

```
INSERT INTO COFFEE
VALUES (1, 03, 2, 'Mezcla', 20.00),
        (2, 01, 1, 'Smooth Blend', 12.00),
        (3, 01, 1, 'Black Brew', 17.00)
;
```

2b. SQL commands and database server response for populating tables.



Query successfully executed in 19ms

```
  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)
);

INSERT INTO COFFEE_SHOP
VALUES (01,'Waynes Coffee Emporium', 'Atlanta', 'GA'),
       (02,'Morning Brew', 'Seattle', 'WA'),
       (03,'Wake Up Cafe', 'Memphis', 'TN')
;

INSERT INTO EMPLOYEE
VALUES (001, 'Peter', 'Parker', '2002-04-29', 'cashier', 02),
       (002, 'Bucky', 'Barnes', '2011-07-19', 'barista', 02),
       (003, 'Bruce','Banner', '2003-06-20', 'manager', 01)
;

INSERT INTO SUPPLIER
VALUES (1, 'Folgers', 'United Sates', 'Jessica Bateman', 'jessica.bateman@gmail.com'),
       (2, 'Cafe De Columbia', 'Columbia', 'Bill Faningham', 'Bfaningham@cafecolumbia.org'),
       (3, 'Keurig', 'United States', 'Brooke Johnson', 'Brooke.Johnson91@outlook.coom')
;

INSERT INTO COFFEE
VALUES (1, 03, 2, 'Mezcla', 20.00),
       (2, 01, 1, 'Smooth Blend', 12.00),
       (3, 01, 1, 'Black Brew', 17.00)
;
```

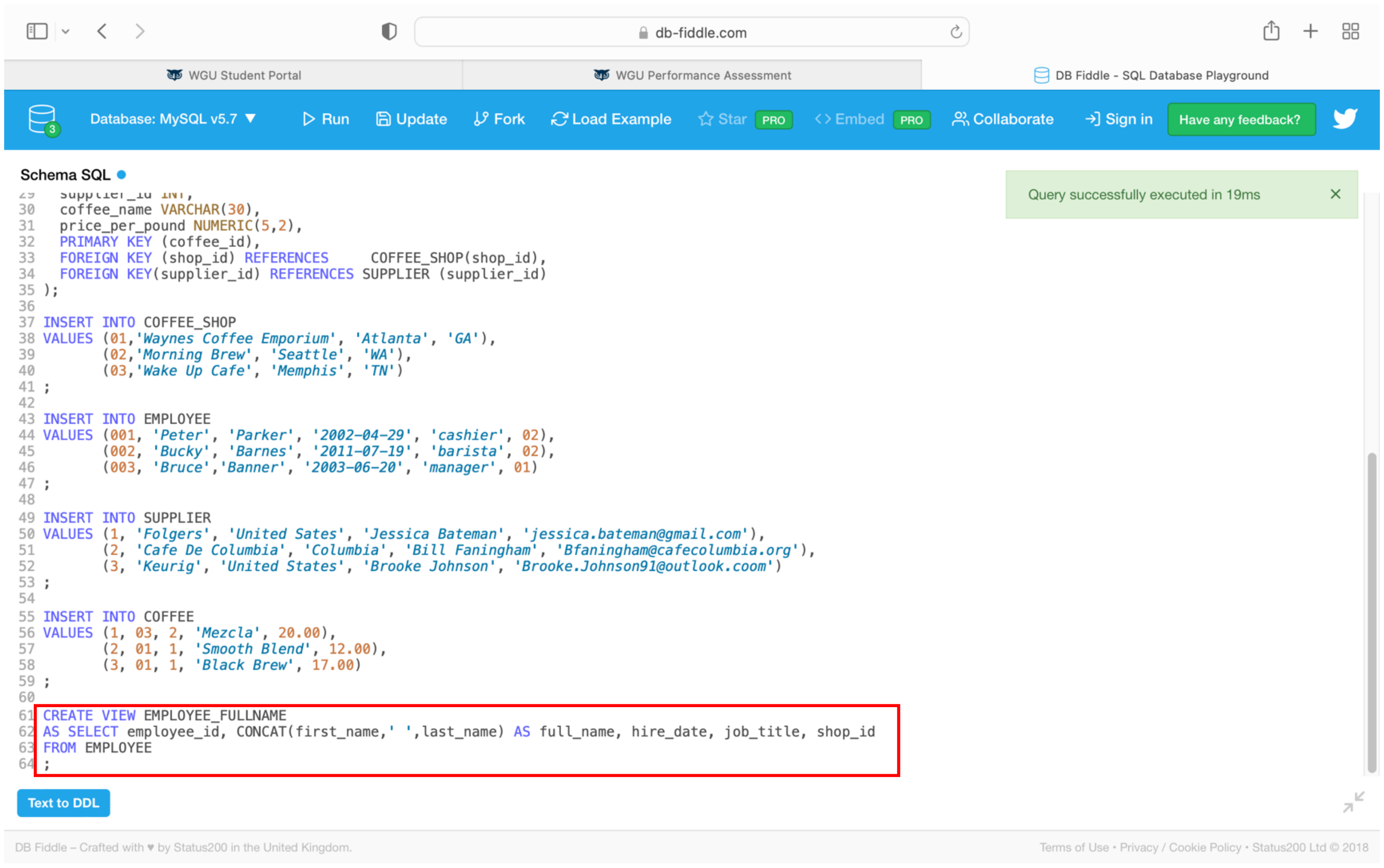
DB Fiddle – Crafted with ♥ by Status200 in the United Kingdom.

Terms of Use • Privacy / Cookie Policy • Status200 Ltd © 2018

3a. SQL code to create a view.

```
CREATE VIEW EMPLOYEE_FULLNAME
AS SELECT employee_id, CONCAT(first_name,' ',last_name) AS full_name, hire_date, job_title, shop_id
FROM EMPLOYEE
;
```

3b. Screenshot of SQL commands and server reponse for creating view.


db-fiddle.com
WGU Student Portal
WGU Performance Assessment
DB Fiddle - SQL Database Playground
Database: MySQL v5.7
Run
Update
Fork
Load Example
Star PRO
Embed PRO
Collaborate
Sign in
Have any feedback?
Schema SQL
Text to DDL
DB Fiddle – Crafted with ♥ by Status200 in the United Kingdom.
Terms of Use • Privacy / Cookie Policy • Status200 Ltd © 2018

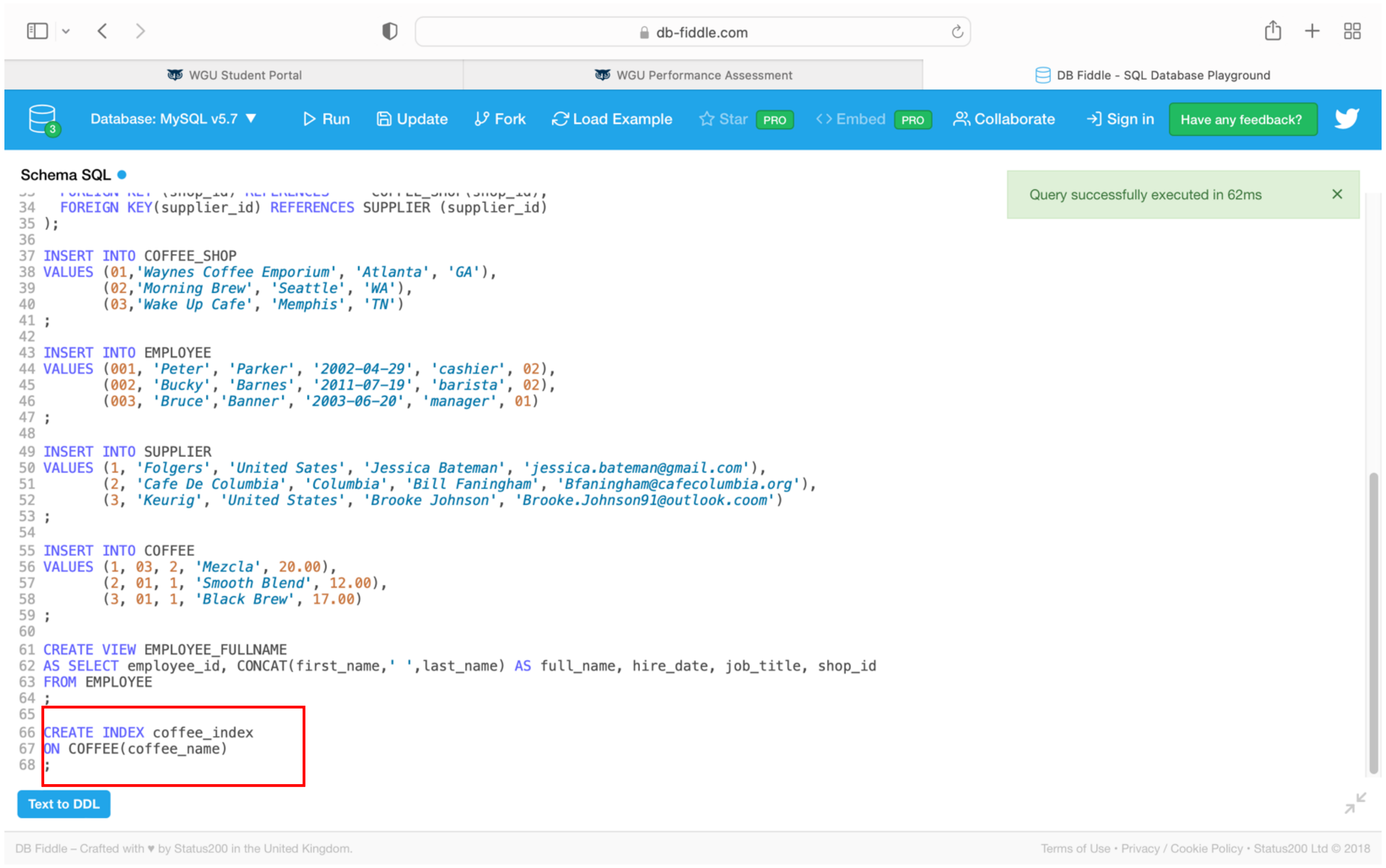
Query successfully executed in 19ms

```
    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)
);

INSERT INTO COFFEE_SHOP
VALUES (01,'Waynes Coffee Emporium', 'Atlanta', 'GA'),
       (02,'Morning Brew', 'Seattle', 'WA'),
       (03,'Wake Up Cafe', 'Memphis', 'TN')
;

INSERT INTO EMPLOYEE
VALUES (001, 'Peter', 'Parker', '2002-04-29', 'cashier', 02),
       (002, 'Bucky', 'Barnes', '2011-07-19', 'barista', 02),
       (003, 'Bruce','Banner', '2003-06-20', 'manager', 01)
;

INSERT INTO SUPPLIER
VALUES (1, 'Folgers', 'United Sates', 'Jessica Bateman', 'jessica.bateman@gmail.com'),
       (2, 'Cafe De Columbia', 'Columbia', 'Bill Faningham', 'Bfaningham@cafecolumbia.org'),
       (3, 'Keurig', 'United States', 'Brooke Johnson', 'Brooke.Johnson91@outlook.coom')
;

INSERT INTO COFFEE
VALUES (1, 03, 2, 'Mezcla', 20.00),
       (2, 01, 1, 'Smooth Blend', 12.00),
       (3, 01, 1, 'Black Brew', 17.00)
;

CREATE VIEW EMPLOYEE_FULLNAME
AS SELECT employee_id, CONCAT(first_name,' ',last_name) AS full_name, hire_date, job_title, shop_id
FROM EMPLOYEE
;
```

4a. SQL code to create and index on the coffee_name field from the 'COFFEE' table.

```
CREATE INDEX coffee_index
ON COFFEE(coffee_name)
;
```

4b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server's response to create index.


db-fiddle.com
WGU Student Portal
WGU Performance Assessment
DB Fiddle - SQL Database Playground
Database: MySQL v5.7
Run
Update
Fork
Load Example
Star PRO
Embed PRO
Collaborate
Sign in
Have any feedback?
Schema SQL
Text to DDL
DB Fiddle – Crafted with ♥ by Status200 in the United Kingdom.
Terms of Use • Privacy / Cookie Policy • Status200 Ltd © 2018

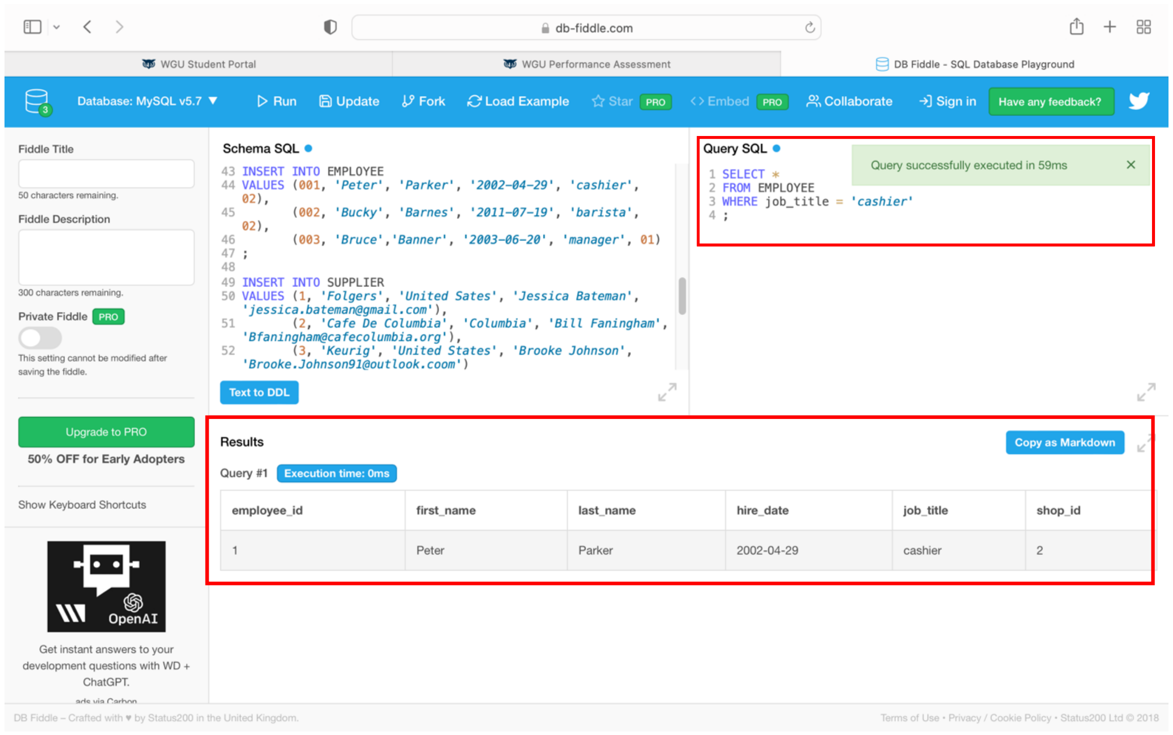
Query successfully executed in 62ms

```
  FOREIGN KEY(shop_id) REFERENCES COFFEE_SHOP(shop_id),
  FOREIGN KEY(supplier_id) REFERENCES SUPPLIER (supplier_id)
);

INSERT INTO COFFEE_SHOP
VALUES (01,'Waynes Coffee Emporium', 'Atlanta', 'GA'),
       (02,'Morning Brew', 'Seattle', 'WA'),
       (03,'Wake Up Cafe', 'Memphis', 'TN')
;

INSERT INTO EMPLOYEE
VALUES (001, 'Peter', 'Parker', '2002-04-29', 'cashier', 02),
       (002, 'Bucky', 'Barnes', '2011-07-19', 'barista', 02),
       (003, 'Bruce','Banner', '2003-06-20', 'manager', 01)
;

INSERT INTO SUPPLIER
VALUES (1, 'Folgers', 'United Sates', 'Jessica Bateman', 'jessica.bateman@gmail.com'),
       (2, 'Cafe De Columbia', 'Columbia', 'Bill Faningham', 'Bfaningham@cafecolumbia.org'),
       (3, 'Keurig', 'United States', 'Brooke Johnson', 'Brooke.Johnson91@outlook.coom')
;

INSERT INTO COFFEE
VALUES (1, 03, 2, 'Mezcla', 20.00),
       (2, 01, 1, 'Smooth Blend', 12.00),
       (3, 01, 1, 'Black Brew', 17.00)
;

CREATE VIEW EMPLOYEE_FULLNAME
AS SELECT employee_id, CONCAT(first_name,' ',last_name) AS full_name, hire_date, job_title, shop_id
FROM EMPLOYEE
;

CREATE INDEX coffee_index
ON COFFEE(coffee_name)
;
```

5a. Provide the SQL code you wrote to create your SFW query.

```
SELECT *
FROM EMPLOYEE
WHERE job_title = 'cashier'
;
```

5b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server's response to SFW.


db-fiddle.com
WGU Student Portal
WGU Performance Assessment
DB Fiddle - SQL Database Playground
Database: MySQL v5.7
Run
Update
Fork
Load Example
Star PRO
Embed PRO
Collaborate
Sign in
Have any feedback?
Fiddle Title
50 characters remaining.
Fiddle Description
300 characters remaining.
Private Fiddle PRO
This setting cannot be modified after saving the fiddle.
Upgrade to PRO
50% OFF for Early Adopters
Show Keyboard Shortcuts
OpenAI
Get instant answers to your development questions with WD + ChatGPT.
ads via Carbon
Schema SQL
Text to DDL
Query SQL
Copy as Markdown
DB Fiddle – Crafted with ♥ by Status200 in the United Kingdom.
Terms of Use • Privacy / Cookie Policy • Status200 Ltd © 2018

Schema SQL

```
INSERT INTO EMPLOYEE
VALUES (001, 'Peter', 'Parker', '2002-04-29', 'cashier',
02),
       (002, 'Bucky', 'Barnes', '2011-07-19', 'barista',
02),
       (003, 'Bruce','Banner', '2003-06-20', 'manager', 01)
;

INSERT INTO SUPPLIER
VALUES (1, 'Folgers', 'United Sates', 'Jessica Bateman',
'jessica.bateman@gmail.com'),
       (2, 'Cafe De Columbia', 'Columbia', 'Bill Faningham',
'Bfaningham@cafecolumbia.org'),
       (3, 'Keurig', 'United States', 'Brooke Johnson',
'Brooke.Johnson91@outlook.coom')
```

Query SQL

```
SELECT *
FROM EMPLOYEE
WHERE job_title = 'cashier'
;
```

Query successfully executed in 59ms

Results

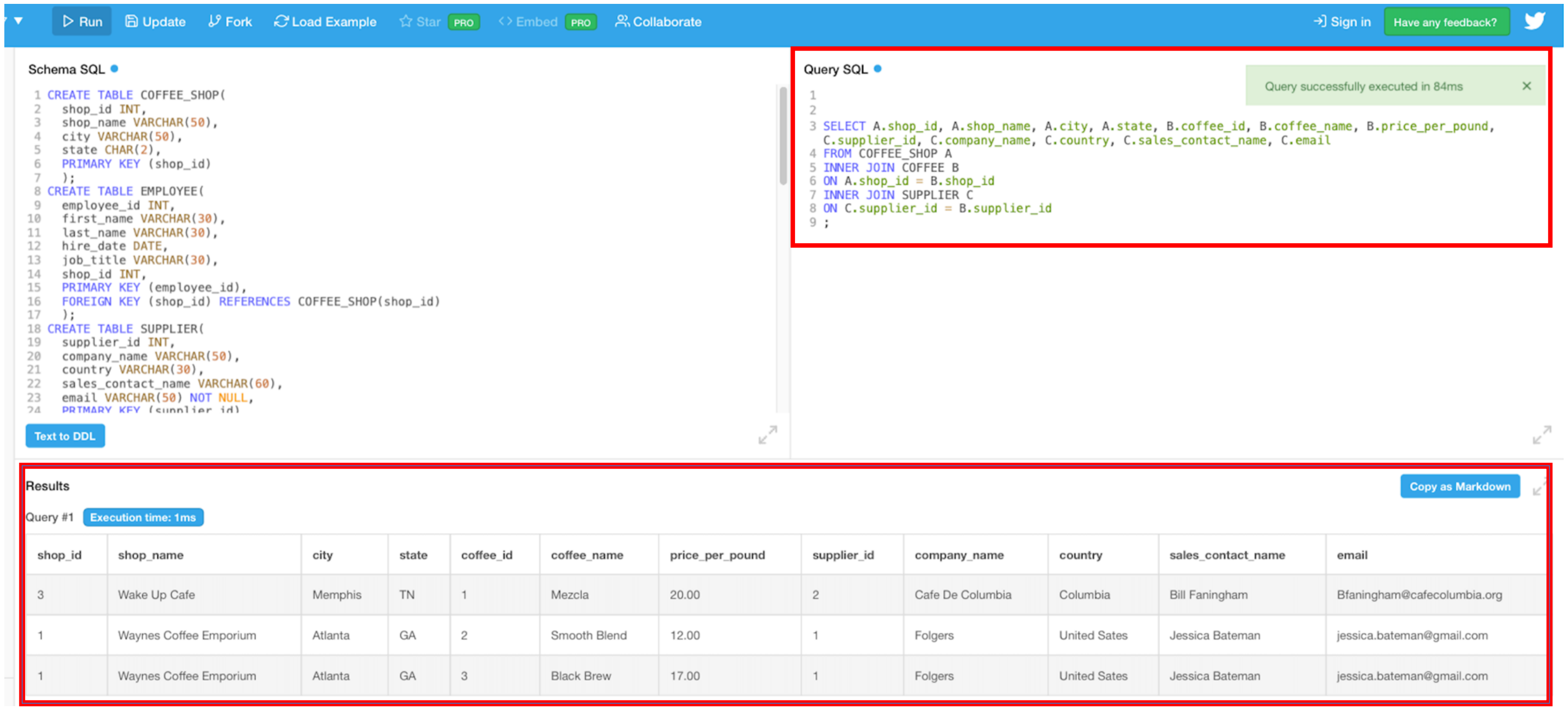
Query #1 Execution time: 0ms

employee_id	first_name	last_name	hire_date	job_title	shop_id
1	Peter	Parker	2002-04-29	cashier	2

6a. Provide the SQL code you wrote to create your table joins query. The query should join three different tables and include attributes from *all* three tables in its output.

```
SELECT A.shop_id, A.shop_name, A.city, A.state, B.coffee_id, B.coffee_name, B.price_per_pound, C.supplier_id, C.company_name, C.country,
C.sales_contact_name, C.email
FROM COFFEE_SHOP A
INNER JOIN COFFEE B
ON A.shop_id = B.shop_id
INNER JOIN SUPPLIER C
ON C.supplier_id = B.supplier_id
;
```

6b. Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server's response.



Schema SQL

```
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)
```

Query SQL

Query successfully executed in 84ms

```


SELECT A.shop_id, A.shop_name, A.city, A.state, B.coffee_id, B.coffee_name, B.price_per_pound,
C.supplier_id, C.company_name, C.country, C.sales_contact_name, C.email
FROM COFFEE_SHOP A
INNER JOIN COFFEE B
ON A.shop_id = B.shop_id
INNER JOIN SUPPLIER C
ON C.supplier_id = B.supplier_id
;
```

Results

Query #1 Execution time: 1ms

shop_id	shop_name	city	state	coffee_id	coffee_name	price_per_pound	supplier_id	company_name	country	sales_contact_name	email
3	Wake Up Cafe	Memphis	TN	1	Mezcla	20.00	2	Cafe De Columbia	Columbia	Bill Faningham	Bfaningham@cafecolumbia.org
1	Waynes Coffee Emporium	Atlanta	GA	2	Smooth Blend	12.00	1	Folgers	United Sates	Jessica Bateman	jessica.bateman@gmail.com
1	Waynes Coffee Emporium	Atlanta	GA	3	Black Brew	17.00	1	Folgers	United Sates	Jessica Bateman	jessica.bateman@gmail.com