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

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
	Delivery Fee
	Special Notes

1:M

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

M:1

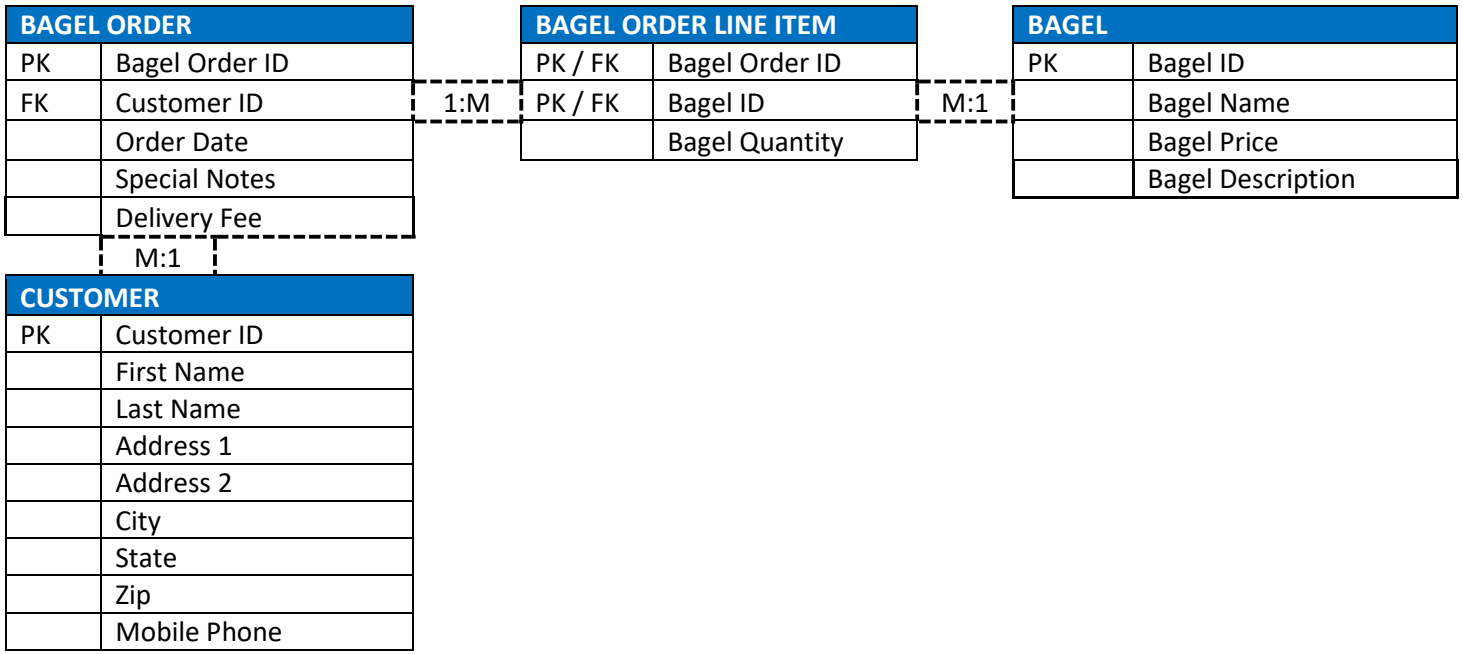
BAGEL	
PK	Bagel Name
FK	Bagel ID
	Bagel Description
	Bagel Price

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-many (1:M), many-to-one (M:1), or many-to-many (M:M).
         1. The Bagel Order Line Item table is an associative table linking the Bagel Order table and the Bagel table together. The Bagel Order Line Item table's composite key was created using the primary key from the first table (Bagel Order ID) and combining it with the primary key in the third table (Bagel ID).
         2. The cardinality between the Bagel Order table and the Bagel Order Line Item table, reading from left to right, is one-to-many (1:M) because at most one bagel order can have many bagel order line items. Conversely, many bagel order line items can only belong to no more than one bagel order.
         3. The cardinality between the Bagel Order Line Item table and the Bagel table, reading from left to right, is many-to-one (M:1) because there can be multiple bagel quantities in the Bagel Order Line Item table for one and only one bagel in the Bagel table. Conversely, one and only one bagel from the Bagel table can be on the Bagel Order Line Item table but with multiple quantities.
      c) Explain how you assigned attributes to the 2NF tables and determined the cardinality of the relationships between your 2NF tables.

1. After organizing the data from Nora's Bagel Bin into 1NF, the information was further separated into multiple tables to achieve 2NF compliance. The original 1NF table had a composite key that consisted of two unique primary keys (Bagel Order ID and Bagel ID). According to 2NF, each non-key attribute must depend on the entire primary key, not just a part of it. The problem with the 1NF table was that the bagel's name, description, and price all depended on the bagel's ID, not the combination of Bagel ID and Bagel Order ID, creating a partial dependency on Bagel ID. The table had to be split into individual tables to make all the attributes functionally dependent on the tables' whole primary key.
2. As discussed in the previous section, the relationship between the Bagel Order table and the Bagel table determined the cardinality. These two entities created a many-to-many (M:M) relationship because many bagel orders could contain many bagels, and conversely, many bagels could be added to many bagel orders. This many-to-many relationship created a problem because we cannot implement an M:M relationship in a relational database because of all the complexities and redundancies it could create. An associative table was introduced to break up the many-to-many relationship, thus creating two one-to-many and many-to-one relationships.

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

Third Normal Form (3NF)



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).
    1. The cardinality for the tables in the third normal form is the same between the Bagel Order table and the Bagel Order Line Item table (1:M). Likewise, the cardinality is the same for the Bagel Order Line Item table and the Bagel table (M:1). The relationship between the Bagel Order table and the Bagel Order Line Item table indicates that one and only one bagel order has many bagel order line items, but many bagel order line items are within one and only one bagel order. The relationship between the Bagel Order Line Item table and the Bagel table signifies that there are multiple quantities of individual bagels listed in the Bagel Order Line Item table, but there can only be one and only one bagel listed from the Bagel table with multiple quantities. The new Customer table has a cardinality of many-to-one (M:1) between the Bagel Order table and itself because one and only one customer can have many bagel orders, but the inverse is not true. Many customers cannot have the same bagel order, but one and only one customer can have many bagel orders.
e) Explain how you assigned attributes to the 3NF tables and determined the cardinality of the relationships between your 3NF tables.
    1. After achieving 2NF, the table entries from Nora’s Bagel Bin were further divided by separating the customer’s information from the Bagel Order table. The third normal form rule states that the table must be in 2NF and that non-key attributes must not be transitively dependent on the prime key, meaning that all non-key attributes must depend solely on the table’s primary key, not on another non-key attribute. The customer’s information (name, address, and phone number) does not depend entirely on the Bagel Order ID but on the customer’s information, thus violating the transitive rule. The customer’s information had to be taken out of the Bagel Order table and placed on its own table.
    2. The new Customer table created a many-to-one (M:1) relationship between the Bagel Order table and the Customer table. The cardinality is many-to-one (M:1) because many bagel orders can belong to one and only one customer, but many customers cannot have the same bagel order.

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

Final Physical Database Model

BAGEL ORDER		
PK	bagel_order_id	INTEGER
FK	customer_id	INTEGER
	order_date	TIMESTAMP
	special_notes	VARCHAR(500)
	delivery_fee	NUMERIC(5,2)

1:M

BAGEL ORDER LINE ITEM		
PK / FK	bagel_order_id	INTEGER
PK / FK	bagel_id	CHAR(3)
	bagel_quantity	INTEGER

M:1

BAGEL		
PK	bagel_id	CHAR(3)
	bagel_name	VARCHAR(50)
	bagel_price	NUMERIC(5,2)
	bagel_description	VARCHAR(100)

M:1

CUSTOMER		
PK	customer_id	INTEGER
	first_name	VARCHAR(40)
	last_name	VARCHAR(40)
	address_1	VARCHAR(40)
	address_2	VARCHAR(40)
	city	VARCHAR(40)
	state	CHAR(2)
	zip	VARCHAR(15)
	mobile_phone	VARCHAR(12)

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.

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

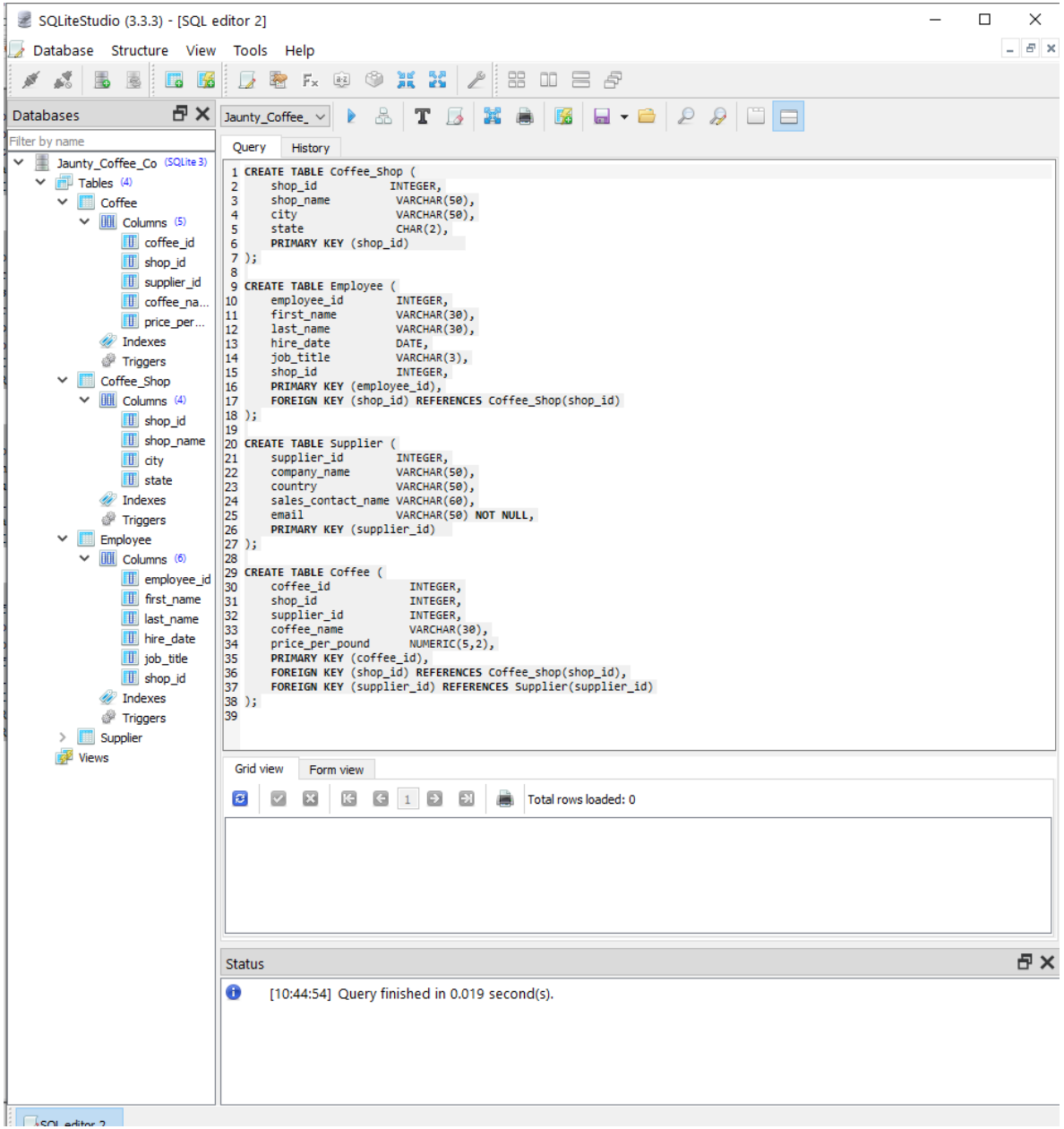
1. Develop SQL code to create each table as specified in the attached "Jaunty Coffee Co. ERD" by doing the following:
   a) Provide the SQL code you wrote to create all the tables.

1.

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

CREATE TABLE Employee (
    employee_id        INTEGER,
    first_name         VARCHAR(30),
    last_name          VARCHAR(30),
    hire_date          DATE,
    job_title          VARCHAR(3),
    shop_id            INTEGER,
    PRIMARY KEY (employee_id),
    FOREIGN KEY (shop_id) REFERENCES Coffee_Shop(shop_id)
);

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

CREATE TABLE Coffee (
    coffee_id            INTEGER,
    shop_id              INTEGER,
    supplier_id          INTEGER,
    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)
);
```

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

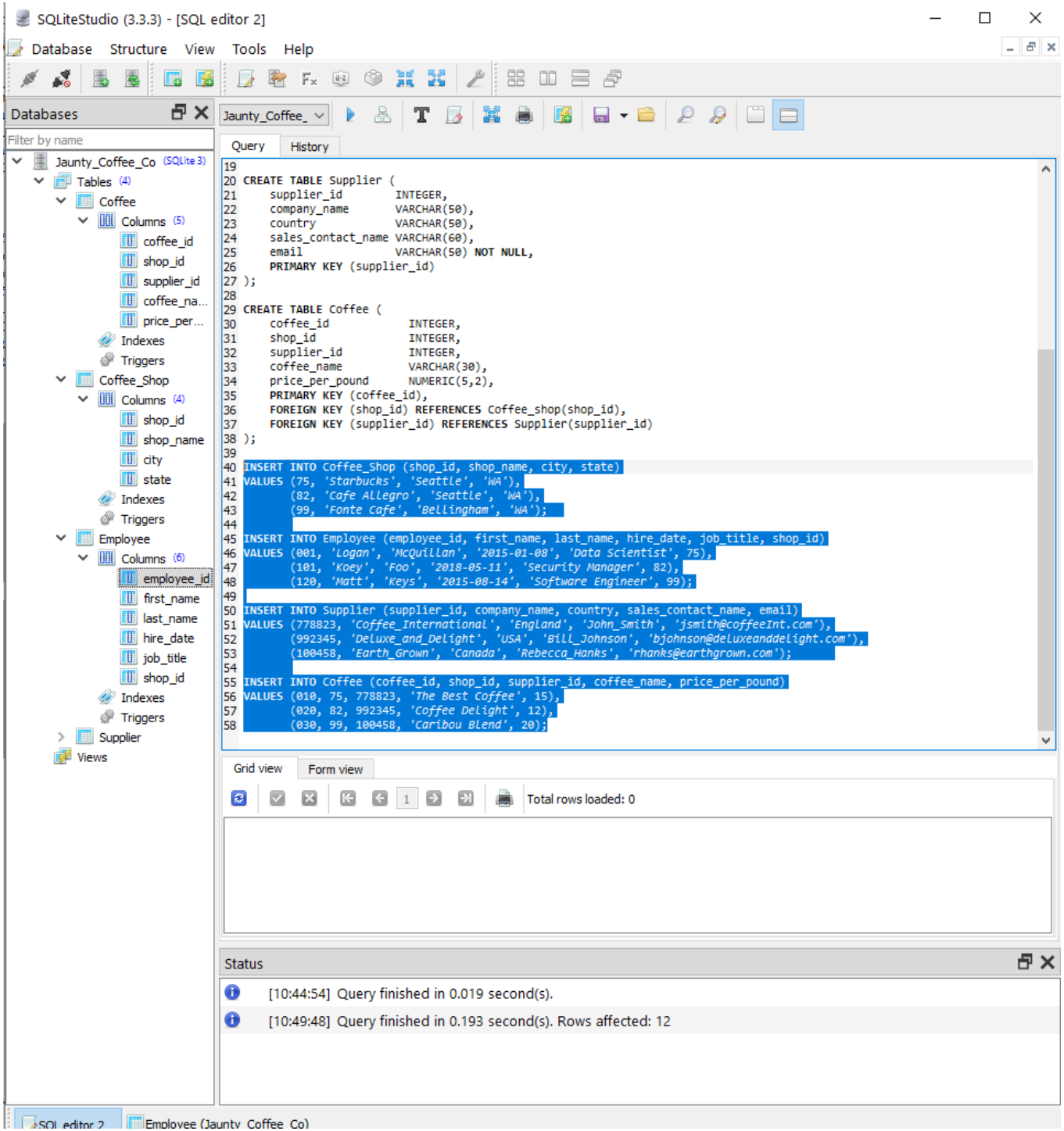
1.



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.

```
INSERT INTO Coffee_Shop (shop_id, shop_name, city, state)
VALUES (75, 'Starbucks', 'Seattle', 'WA'),
       (82, 'Cafe Allegro', 'Seattle', 'WA'),
       (99, 'Fonte Cafe', 'Bellingham', 'WA');

INSERT INTO Employee (employee_id, first_name, last_name, hire_date, job_title, shop_id)
VALUES (001, 'Logan', 'McQuillan', '2015-01-08', 'Data Scientist', 75),
       (101, 'Koey', 'Foo', '2018-05-11', 'Security Manager', 82),
       (120, 'Matt', 'Keys', '2015-08-14', 'Software Engineer', 99);

INSERT INTO Supplier (supplier_id, company_name, country, sales_contact_name, email)
VALUES (778823, 'Coffee_International', 'England', 'John_Smith', 'jsmith@coffeeInt.com'),
       (992345, 'Deluxe_and_Delight', 'USA', 'Bill_Johnson', 'bjohnson@deluxeanddelight.com'),
       (100458, 'Earth_Grown', 'Canada', 'Rebecca_Hanks', 'rhanks@earthgrown.com');

INSERT INTO Coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
VALUES (010, 75, 778823, 'The Best Coffee', 15),
       (020, 82, 992345, 'Coffee Delight', 12),
       (030, 99, 100458, 'Caribou Blend', 20);
```

   1.
   b) Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server's response.


SQLiteStudio (3.3.3) - [SQL editor 2]
Database Structure View Tools Help
Databases
Filter by name
Jaunty_Coffee_Co (SQLite 3)
Tables (4)
Coffee
Columns (5)
coffee_id
shop_id
supplier_id
coffee_na...
price_per...
Indexes
Triggers
Coffee_Shop
Columns (4)
shop_id
shop_name
city
state
Indexes
Triggers
Employee
Columns (6)
employee_id
first_name
last_name
hire_date
job_title
shop_id
Indexes
Triggers
Supplier
Views
Jaunty_Coffee_
Query
History
19
20 CREATE TABLE Supplier (
21 supplier_id INTEGER,
22 company_name VARCHAR(50),
23 country VARCHAR(50),
24 sales_contact_name VARCHAR(60),
25 email VARCHAR(50) NOT NULL,
26 PRIMARY KEY (supplier_id)
27);
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
40 INSERT INTO Coffee_Shop (shop_id, shop_name, city, state)
41 VALUES (75, 'Starbucks', 'Seattle', 'WA'),
42 (82, 'Cafe Allegro', 'Seattle', 'WA'),
43 (99, 'Fonte Cafe', 'Bellingham', 'WA');
44
45 INSERT INTO Employee (employee_id, first_name, last_name, hire_date, job_title, shop_id)
46 VALUES (001, 'Logan', 'McQuillan', '2015-01-08', 'Data Scientist', 75),
47 (101, 'Koey', 'Foo', '2018-05-11', 'Security Manager', 82),
48 (120, 'Matt', 'Keys', '2015-08-14', 'Software Engineer', 99);
49
50 INSERT INTO Supplier (supplier_id, company_name, country, sales_contact_name, email)
51 VALUES (778823, 'Coffee_International', 'England', 'John_Smith', 'jsmith@coffeeInt.com'),
52 (992345, 'Deluxe_and_Delight', 'USA', 'Bill_Johnson', 'bjohnson@deluxeanddelight.com'),
53 (100458, 'Earth_Grown', 'Canada', 'Rebecca_Hanks', 'rhanks@earthgrown.com');
54
55 INSERT INTO Coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
56 VALUES (010, 75, 778823, 'The Best Coffee', 15),
57 (020, 82, 992345, 'Coffee Delight', 12),
58 (030, 99, 100458, 'Caribou Blend', 20);
Grid view
Form view
1
Total rows loaded: 0
Status
[10:44:54] Query finished in 0.019 second(s).
[10:49:48] Query finished in 0.193 second(s). Rows affected: 12
SQL editor 2
Employee (Jaunty_Coffee_Co)

1.

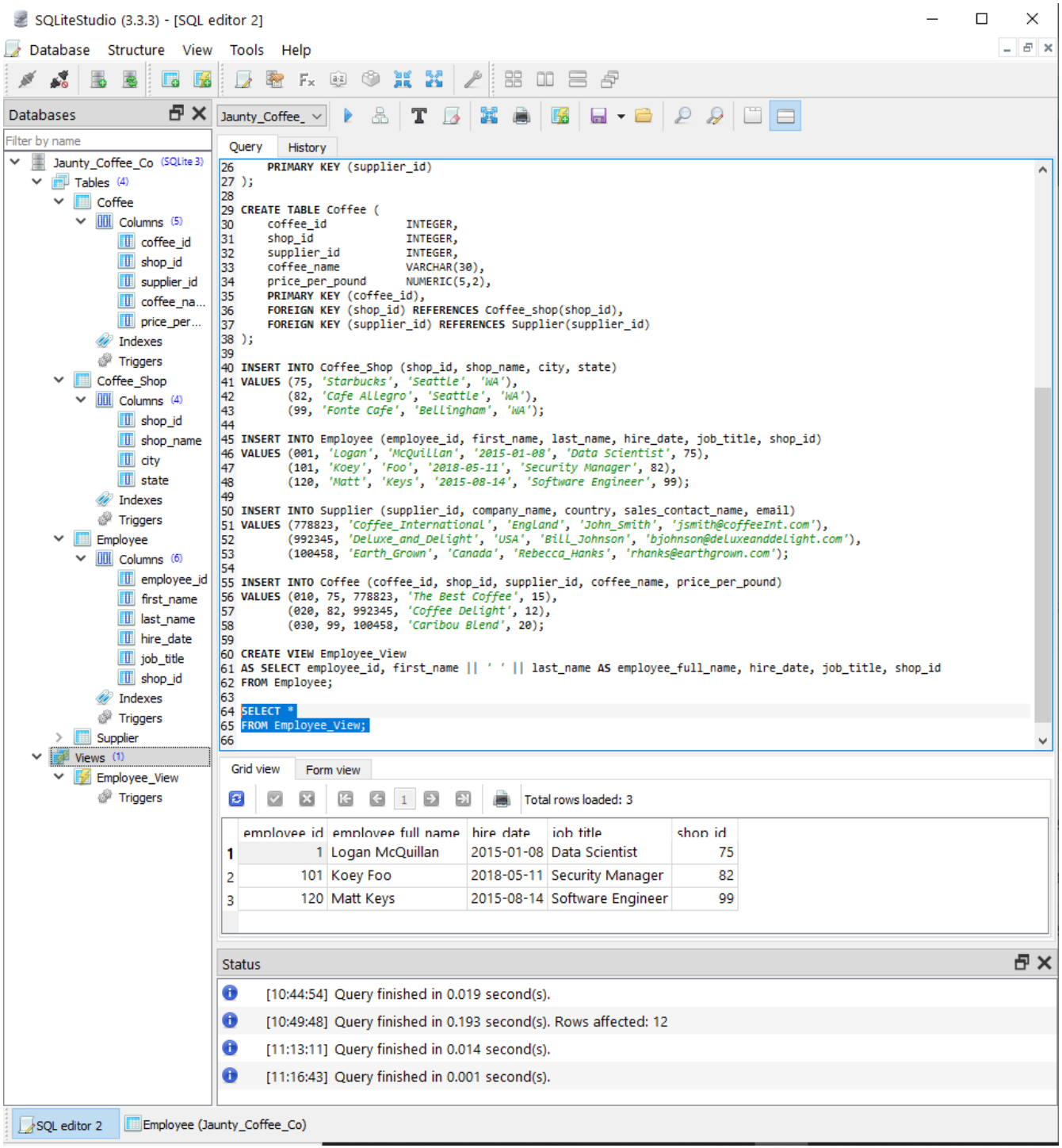
3. Develop SQL code to create a view by doing the following:
   a) Provide the SQL code you wrote to create your view. The view should show all of the information from the “Employee” table but concatenate each employee’s first and last name, formatted with a space between the first and last name, into a new attribute called employee_full_name.

```
CREATE VIEW Employee_View
AS SELECT employee_id, first_name || ' ' || last_name AS employee_full_name, hire_date, job_title, shop_id
FROM Employee;
```

1.

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

1.

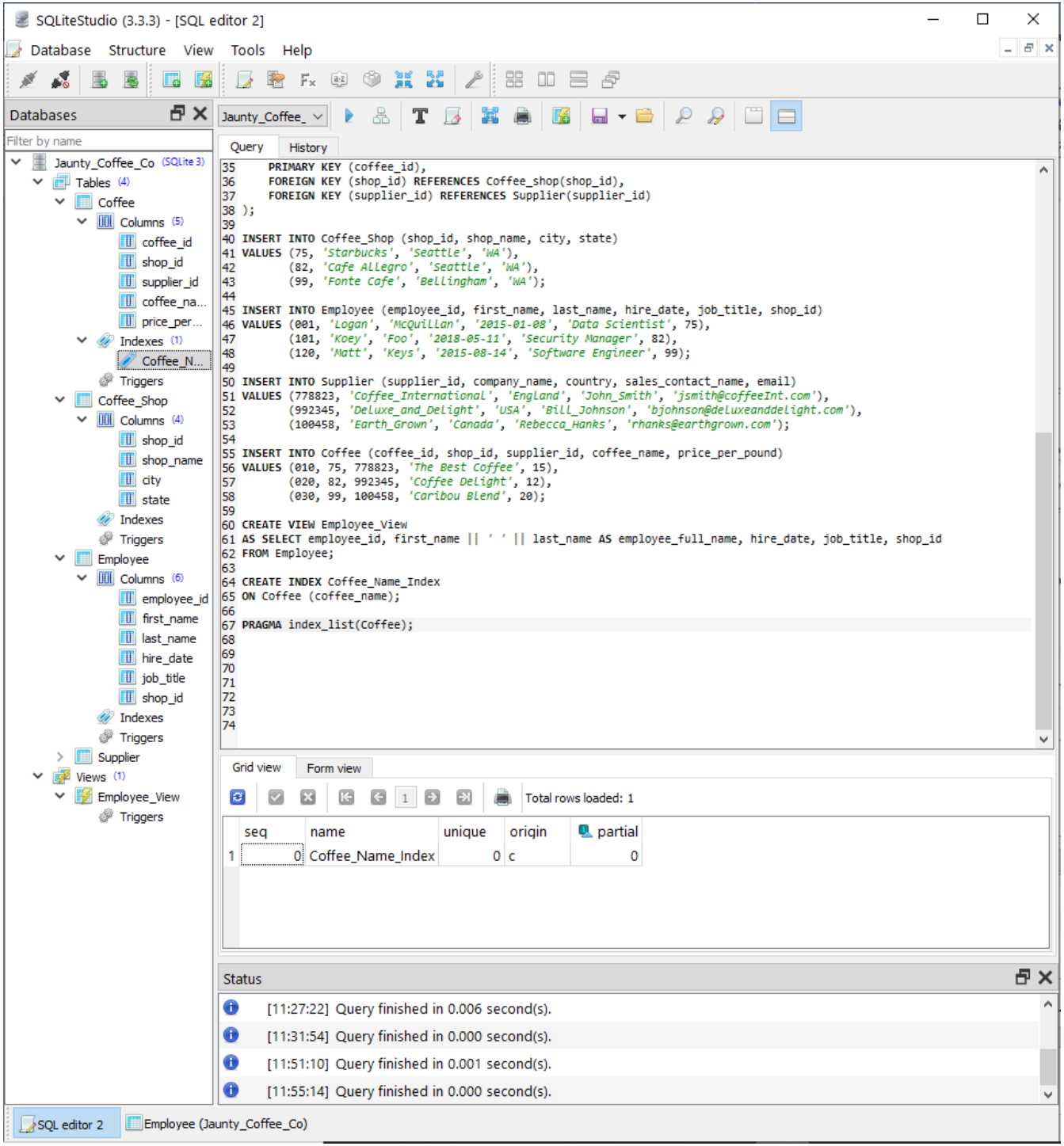

SQLiteStudio (3.3.3) - [SQL editor 2]
Database Structure View Tools Help
Databases
Filter by name
Jaunty_Coffee_Co (SQLite 3)
Tables (4)
Coffee
Columns (5)
coffee_id
shop_id
supplier_id
coffee_na...
price_per...
Indexes
Triggers
Coffee_Shop
Columns (4)
shop_id
shop_name
city
state
Indexes
Triggers
Employee
Columns (6)
employee_id
first_name
last_name
hire_date
job_title
shop_id
Indexes
Triggers
Supplier
Views (1)
Employee_View
Triggers
Jaunty_Coffee_
Query
History
26 PRIMARY KEY (supplier_id)
27);
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
40 INSERT INTO Coffee_Shop (shop_id, shop_name, city, state)
41 VALUES (75, 'Starbucks', 'Seattle', 'WA'),
42 (82, 'Cafe Allegro', 'Seattle', 'WA'),
43 (99, 'Fonte Cafe', 'Bellingham', 'WA');
44
45 INSERT INTO Employee (employee_id, first_name, last_name, hire_date, job_title, shop_id)
46 VALUES (001, 'Logan', 'McQuillan', '2015-01-08', 'Data Scientist', 75),
47 (101, 'Koey', 'Foo', '2018-05-11', 'Security Manager', 82),
48 (120, 'Matt', 'Keys', '2015-08-14', 'Software Engineer', 99);
49
50 INSERT INTO Supplier (supplier_id, company_name, country, sales_contact_name, email)
51 VALUES (778823, 'Coffee_International', 'England', 'John_Smith', 'jsmith@coffeeInt.com'),
52 (992345, 'Deluxe_and_Delight', 'USA', 'Bill_Johnson', 'bjohnson@deluxeanddelight.com'),
53 (100458, 'Earth_Grown', 'Canada', 'Rebecca_Hanks', 'rhanks@earthgrown.com');
54
55 INSERT INTO Coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
56 VALUES (010, 75, 778823, 'The Best Coffee', 15),
57 (020, 82, 992345, 'Coffee Delight', 12),
58 (030, 99, 100458, 'Caribou Blend', 20);
59
60 CREATE VIEW Employee_View
61 AS SELECT employee_id, first_name || ' ' || last_name AS employee_full_name, hire_date, job_title, shop_id
62 FROM Employee;
63
64 SELECT *
65 FROM Employee_View;
66
Grid view
Form view
Total rows loaded: 3
employee_id employee_full_name hire_date job_title shop_id
1 1 Logan McQuillan 2015-01-08 Data Scientist 75
2 101 Koey Foo 2018-05-11 Security Manager 82
3 120 Matt Keys 2015-08-14 Software Engineer 99
Status
[10:44:54] Query finished in 0.019 second(s).
[10:49:48] Query finished in 0.193 second(s). Rows affected: 12
[11:13:11] Query finished in 0.014 second(s).
[11:16:43] Query finished in 0.001 second(s).
SQL editor 2
Employee (Jaunty_Coffee_Co)

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.

```
CREATE INDEX Coffee_Name_Index
ON Coffee (coffee_name);


PRAGMA index_list(Coffee);
```

   1.
   b) Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server’s response.

1.

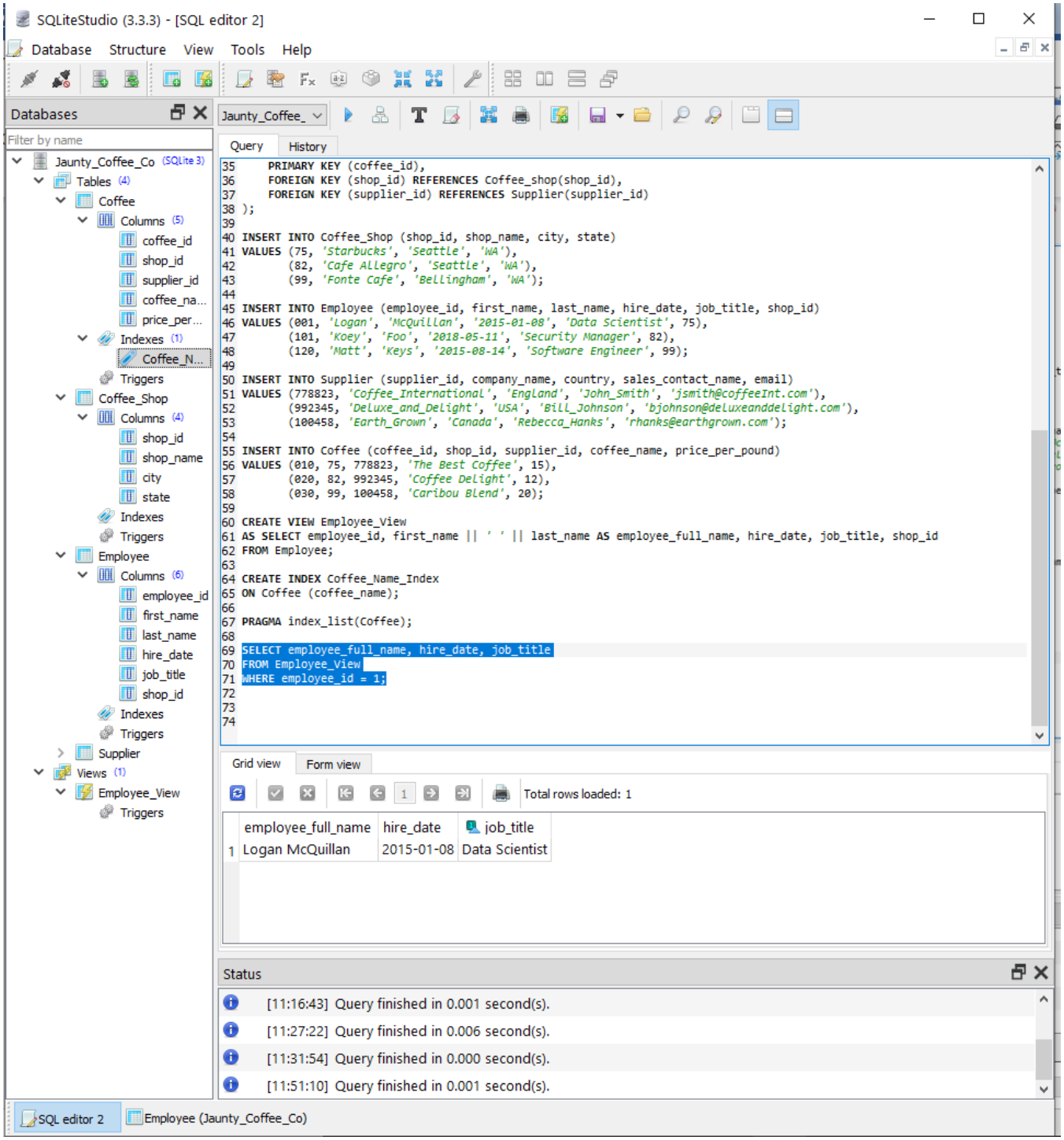

SQLiteStudio (3.3.3) - [SQL editor 2]
Database Structure View Tools Help
Databases
Filter by name
Jaunty_Coffee_Co (SQLite 3)
Tables (4)
Coffee
Columns (5)
coffee_id
shop_id
supplier_id
coffee_na...
price_per...
Indexes (1)
Coffee_N...
Triggers
Coffee_Shop
Columns (4)
shop_id
shop_name
city
state
Indexes
Triggers
Employee
Columns (6)
employee_id
first_name
last_name
hire_date
job_title
shop_id
Indexes
Triggers
Supplier
Views (1)
Employee_View
Triggers
Jaunty_Coffee_
Query
History
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
40 INSERT INTO Coffee_Shop (shop_id, shop_name, city, state)
41 VALUES (75, 'Starbucks', 'Seattle', 'WA'),
42 (82, 'Cafe Allegro', 'Seattle', 'WA'),
43 (99, 'Fonte Cafe', 'Bellingham', 'WA');
44
45 INSERT INTO Employee (employee_id, first_name, last_name, hire_date, job_title, shop_id)
46 VALUES (001, 'Logan', 'McQuillan', '2015-01-08', 'Data Scientist', 75),
47 (101, 'Koey', 'Foo', '2018-05-11', 'Security Manager', 82),
48 (120, 'Matt', 'Keys', '2015-08-14', 'Software Engineer', 99);
49
50 INSERT INTO Supplier (supplier_id, company_name, country, sales_contact_name, email)
51 VALUES (778823, 'Coffee_International', 'England', 'John_Smith', 'jsmith@coffeeInt.com'),
52 (992345, 'Deluxe_and_Delight', 'USA', 'Bill_Johnson', 'bjohnson@deluxeanddelight.com'),
53 (100458, 'Earth_Grown', 'Canada', 'Rebecca_Hanks', 'rhanks@earthgrown.com');
54
55 INSERT INTO Coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
56 VALUES (010, 75, 778823, 'The Best Coffee', 15),
57 (020, 82, 992345, 'Coffee Delight', 12),
58 (030, 99, 100458, 'Caribou Blend', 20);
59
60 CREATE VIEW Employee_View
61 AS SELECT employee_id, first_name || ' ' || last_name AS employee_full_name, hire_date, job_title, shop_id
62 FROM Employee;
63
64 CREATE INDEX Coffee_Name_Index
65 ON Coffee (coffee_name);
66
67 PRAGMA index_list(Coffee);
68
69
70
71
72
73
74
Grid view
Form view
Total rows loaded: 1
seq name unique origin partial
1 0 Coffee_Name_Index 0 c 0
Status
[11:27:22] Query finished in 0.006 second(s).
[11:31:54] Query finished in 0.000 second(s).
[11:51:10] Query finished in 0.001 second(s).
[11:55:14] Query finished in 0.000 second(s).
SQL editor 2
Employee (Jaunty_Coffee_Co)

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.

```
SELECT employee_full_name, hire_date, job_title
FROM Employee_View
WHERE employee_id = 1;
```

   1.
   b) Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server's response.

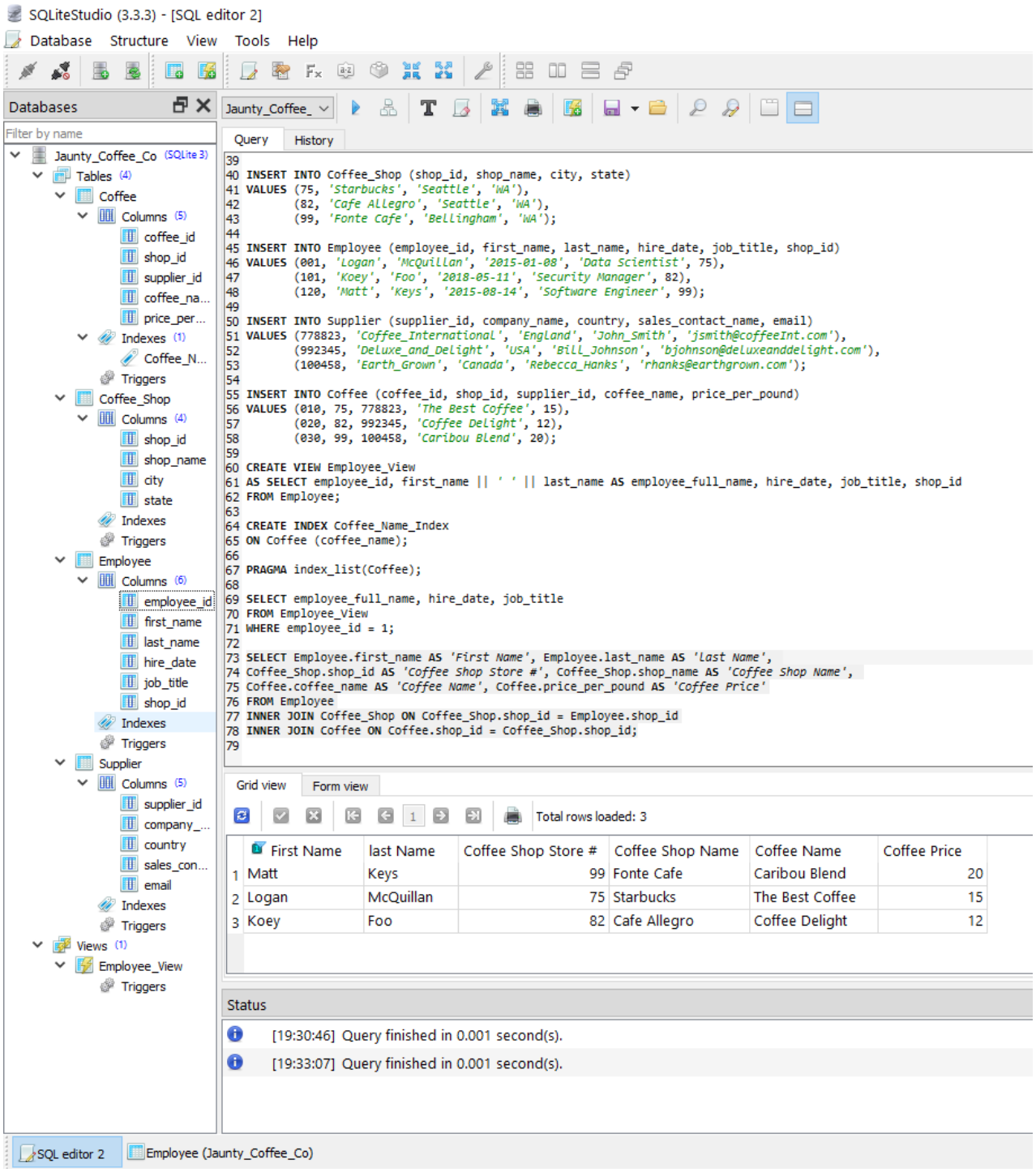
1.


SQLiteStudio (3.3.3) - [SQL editor 2]
Database Structure View Tools Help
Databases
Filter by name
Jaunty_Coffee_Co (SQLite 3)
Tables (4)
Coffee
Columns (5)
coffee_id
shop_id
supplier_id
coffee_na...
price_per...
Indexes (1)
Coffee_N...
Triggers
Coffee_Shop
Columns (4)
shop_id
shop_name
city
state
Indexes
Triggers
Employee
Columns (6)
employee_id
first_name
last_name
hire_date
job_title
shop_id
Indexes
Triggers
Supplier
Views (1)
Employee_View
Triggers
Jaunty_Coffee_
Query
History
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
40 INSERT INTO Coffee_Shop (shop_id, shop_name, city, state)
41 VALUES (75, 'Starbucks', 'Seattle', 'WA'),
42 (82, 'Cafe Allegro', 'Seattle', 'WA'),
43 (99, 'Fonte Cafe', 'Bellingham', 'WA');
44
45 INSERT INTO Employee (employee_id, first_name, last_name, hire_date, job_title, shop_id)
46 VALUES (001, 'Logan', 'McQuillan', '2015-01-08', 'Data Scientist', 75),
47 (101, 'Koey', 'Foo', '2018-05-11', 'Security Manager', 82),
48 (120, 'Matt', 'Keys', '2015-08-14', 'Software Engineer', 99);
49
50 INSERT INTO Supplier (supplier_id, company_name, country, sales_contact_name, email)
51 VALUES (778823, 'Coffee_International', 'England', 'John_Smith', 'jsmith@coffeeInt.com'),
52 (992345, 'Deluxe_and_Delight', 'USA', 'Bill_Johnson', 'bjohnson@deluxeanddelight.com'),
53 (100458, 'Earth_Grown', 'Canada', 'Rebecca_Hanks', 'rhanks@earthgrown.com');
54
55 INSERT INTO Coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
56 VALUES (010, 75, 778823, 'The Best Coffee', 15),
57 (020, 82, 992345, 'Coffee Delight', 12),
58 (030, 99, 100458, 'Caribou Blend', 20);
59
60 CREATE VIEW Employee_View
61 AS SELECT employee_id, first_name || ' ' || last_name AS employee_full_name, hire_date, job_title, shop_id
62 FROM Employee;
63
64 CREATE INDEX Coffee_Name_Index
65 ON Coffee (coffee_name);
66
67 PRAGMA index_list(Coffee);
68
69 SELECT employee_full_name, hire_date, job_title
70 FROM Employee_View
71 WHERE employee_id = 1;
72
73
74
Grid view
Form view
Total rows loaded: 1
employee_full_name hire_date job_title
1 Logan McQuillan 2015-01-08 Data Scientist
Status
[11:16:43] Query finished in 0.001 second(s).
[11:27:22] Query finished in 0.006 second(s).
[11:31:54] Query finished in 0.000 second(s).
[11:51:10] Query finished in 0.001 second(s).
SQL editor 2
Employee (Jaunty_Coffee_Co)

6. Develop SQL code to create a query by doing the following:
   a) Provide the SQL code you wrote to crate your table joins query. The query should join together three different tables and include attributes from all three tables in its output.

```
SELECT Employee.first_name AS 'First Name', Employee.last_name AS 'last Name',
Coffee_Shop.shop_id AS 'Coffee Shop Store #', Coffee_Shop.shop_name AS 'Coffee Shop Name',
Coffee.coffee_name AS 'Coffee Name', Coffee.price_per_pound AS 'Coffee Price'
FROM Employee
INNER JOIN Coffee_Shop ON Coffee_Shop.shop_id = Employee.shop_id
INNER JOIN Coffee ON Coffee.shop_id = Coffee_Shop.shop_id;
```

   1.
   b) Demonstrate that you tested your code by providing a screenshot showing your SQL commands and the database server's response.


SQLiteStudio (3.3.3) - [SQL editor 2]
Database Structure View Tools Help
Databases
Filter by name
Jaunty_Coffee_Co (SQLite 3)
Tables (4)
Coffee
Columns (5)
coffee_id
shop_id
supplier_id
coffee_na...
price_per...
Indexes (1)
Coffee_N...
Triggers
Coffee_Shop
Columns (4)
shop_id
shop_name
city
state
Indexes
Triggers
Employee
Columns (6)
employee_id
first_name
last_name
hire_date
job_title
shop_id
Indexes
Triggers
Supplier
Columns (5)
supplier_id
company_...
country
sales_con...
email
Indexes
Triggers
Views (1)
Employee_View
Triggers
Jaunty_Coffee_
Query
History
39
40 INSERT INTO Coffee_Shop (shop_id, shop_name, city, state)
41 VALUES (75, 'Starbucks', 'Seattle', 'WA'),
42 (82, 'Cafe Allegro', 'Seattle', 'WA'),
43 (99, 'Fonte Cafe', 'Bellingham', 'WA');
44
45 INSERT INTO Employee (employee_id, first_name, last_name, hire_date, job_title, shop_id)
46 VALUES (001, 'Logan', 'McQuillan', '2015-01-08', 'Data Scientist', 75),
47 (101, 'Koey', 'Foo', '2018-05-11', 'Security Manager', 82),
48 (120, 'Matt', 'Keys', '2015-08-14', 'Software Engineer', 99);
49
50 INSERT INTO Supplier (supplier_id, company_name, country, sales_contact_name, email)
51 VALUES (778823, 'Coffee_International', 'England', 'John_Smith', 'jsmith@coffeeInt.com'),
52 (992345, 'DeLuxe_and_Delight', 'USA', 'Bill_Johnson', 'bjohnson@deluxeanddelight.com'),
53 (100458, 'Earth_Grown', 'Canada', 'Rebecca_Hanks', 'rhanks@earthgrown.com');
54
55 INSERT INTO Coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
56 VALUES (010, 75, 778823, 'The Best Coffee', 15),
57 (020, 82, 992345, 'Coffee Delight', 12),
58 (030, 99, 100458, 'Caribou Blend', 20);
59
60 CREATE VIEW Employee_View
61 AS SELECT employee_id, first_name || ' ' || last_name AS employee_full_name, hire_date, job_title, shop_id
62 FROM Employee;
63
64 CREATE INDEX Coffee_Name_Index
65 ON Coffee (coffee_name);
66
67 PRAGMA index_list(Coffee);
68
69 SELECT employee_full_name, hire_date, job_title
70 FROM Employee_View
71 WHERE employee_id = 1;
72
73 SELECT Employee.first_name AS 'First Name', Employee.last_name AS 'Last Name',
74 Coffee_Shop.shop_id AS 'Coffee Shop Store #', Coffee_Shop.shop_name AS 'Coffee Shop Name',
75 Coffee.coffee_name AS 'Coffee Name', Coffee.price_per_pound AS 'Coffee Price'
76 FROM Employee
77 INNER JOIN Coffee_Shop ON Coffee_Shop.shop_id = Employee.shop_id
78 INNER JOIN Coffee ON Coffee.shop_id = Coffee_Shop.shop_id;
79
Grid view
Form view
Total rows loaded: 3
First Name | last Name | Coffee Shop Store # | Coffee Shop Name | Coffee Name | Coffee Price
1 Matt | Keys | 99 | Fonte Cafe | Caribou Blend | 20
2 Logan | McQuillan | 75 | Starbucks | The Best Coffee | 15
3 Koey | Foo | 82 | Cafe Allegro | Coffee Delight | 12
Status
[19:30:46] Query finished in 0.001 second(s).
[19:33:07] Query finished in 0.001 second(s).
SQL editor 2
Employee (Jaunty_Coffee_Co)

1.