Part A

Nora’s Bagel Bin Database Blueprints

First Normal Form (1NF)

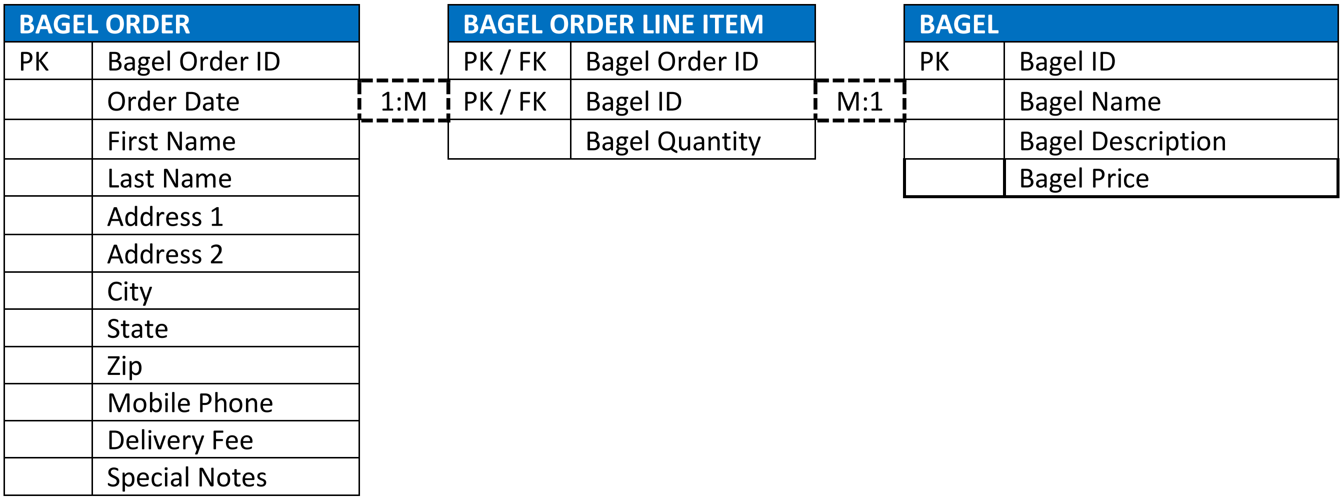
BAGEL ORDER	
PK	Bagel Order ID
PK	Bagel ID
	Order Date
	First Name
	Last Name
	Address 1
	Address 2
	City
	State
	Zip
	Mobile Phone
	Delivery Fee
	Bagel Name
	Bagel Description
	Bagel Price
	Bagel Quantity
	Special Notes

A.1

a. & b.

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

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

BAGEL	
PK	Bagel ID
	Bagel Name
	Bagel Description
	Bagel Price

c.

Second normal form requires that each non-key attribute depend on the whole primary key, not just part. This is to prevent things like insertion anomalies, deletion anomalies, and update anomalies.

In the first normal form table, there are items that depend on one part of the primary key but not the other.

Ex:

Delivery Fee depends on **Bagel Order ID** but not **Bagel ID**, and **Bagel Name** depends on **Bagel ID** but not **Bagel Order ID.**

As for **Bagel Quantity**, it still depends on both **Bagel Order ID** and **Bagel ID**, so it gets its own table.

Relationship between ‘Bagel Order’ and ‘Bagel Order Line Item’:

One to Many. One bagel order might have many line items, but each line item can only belong to one order.

Relationship between ‘Bagel Order Line Item’ and ‘Bagle’:

Many to One. Each line item can only have one (type of) bagel, but a bagel might belong to many line items.

A.2

a. b. c. & d.

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

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

e.

In the second normal form relation, the ‘Bagel Order’ table has functional dependencies. For example, **City** and **State** are functionally dependent on **Address 1** and/or **Address 2** as well as on **First Name** and **Last Name** combined. To fix this and achieve third normal form, we add a new ‘Customer’ table. We connect the ‘Customer’ table with the ‘Bagel Order’ table by creating a primary key **Customer ID** in the ‘Customer’ table that acts as a foreign key in the ‘Bagel Order’ table.

Relationship between ‘Bagel Order’ and ‘Customer’:

Many to One. Each bagel order may belong to at most one customer. One customer can have many orders.

A.3

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

Final Physical Database Model

BAGEL ORDER		
PK	bagel_order_ID	INT
FK	customer_ID	INT
	order_date	TIMESTAMP
	delivery_fee	NUMERIC(5,2)
	special_notes	VARCHAR(200)

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(200)
	bagel_price	NUMERIC(7,2)

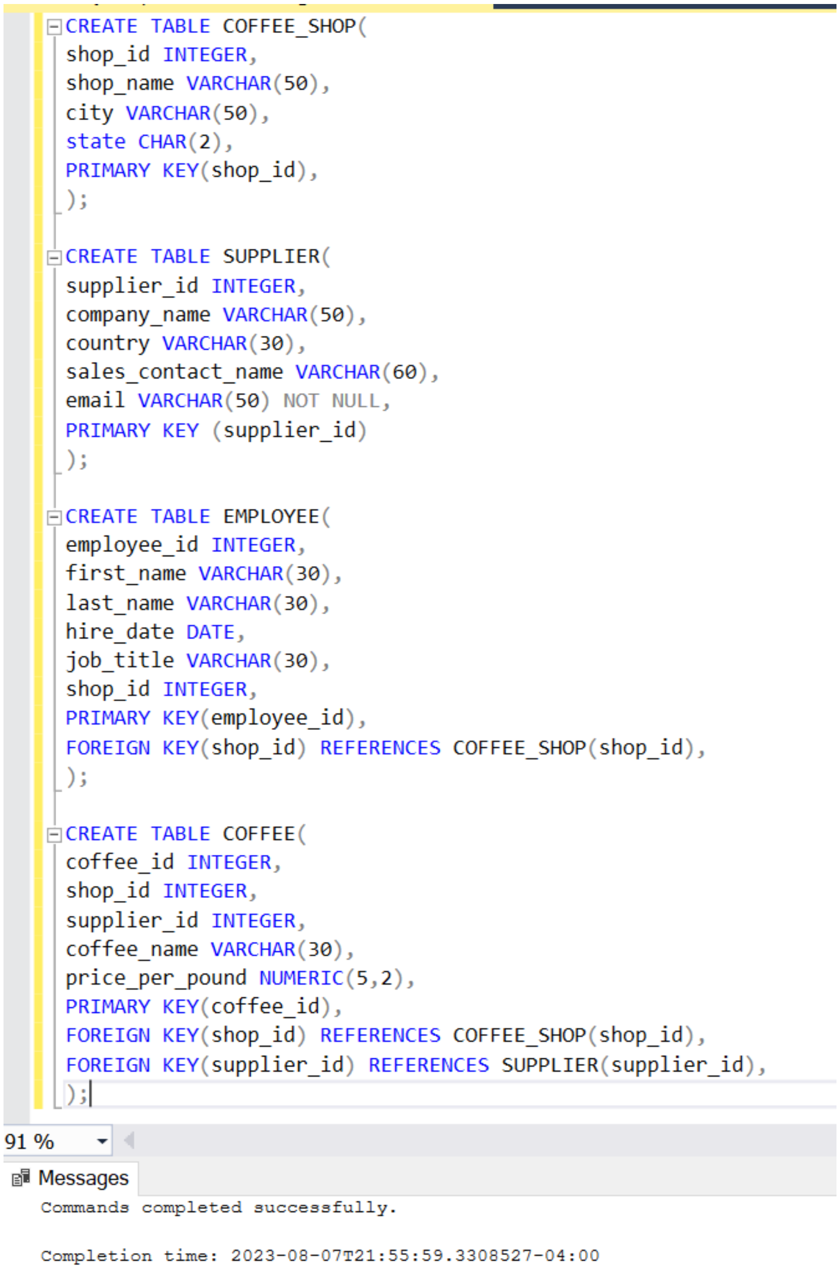
M:1

CUSTOMER		
PK	customer_ID	INT
	first_name	CHAR(50)
	last_name	CHAR(50)
	address_1	CHAR(100)
	address_2	CHAR(100)
	city	CHAR(50)
	state	CHAR(18)
	zip	INT
	mobile_phone	INT

Part B

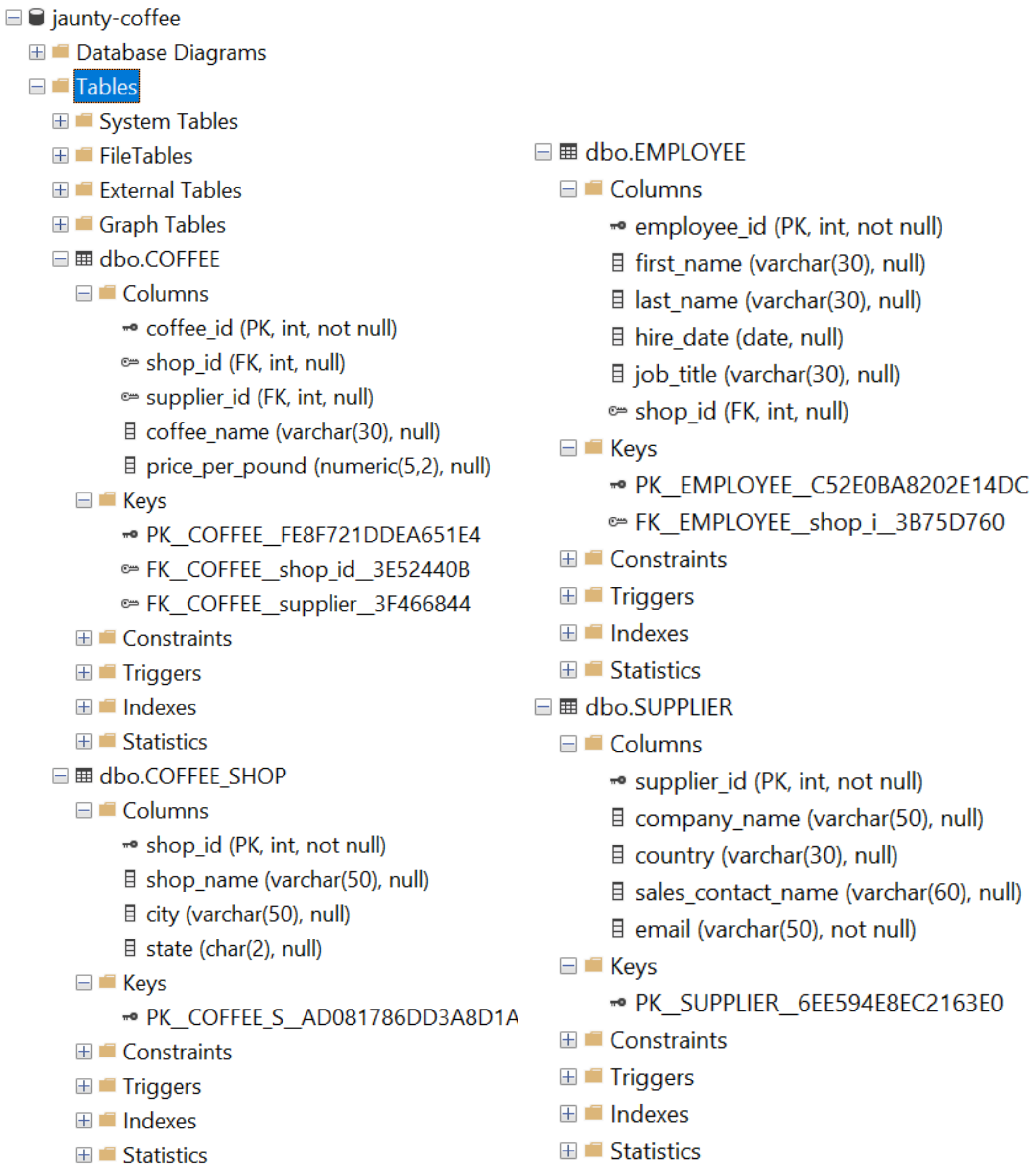
(using Microsoft SQL Server Management Studio)

B.1



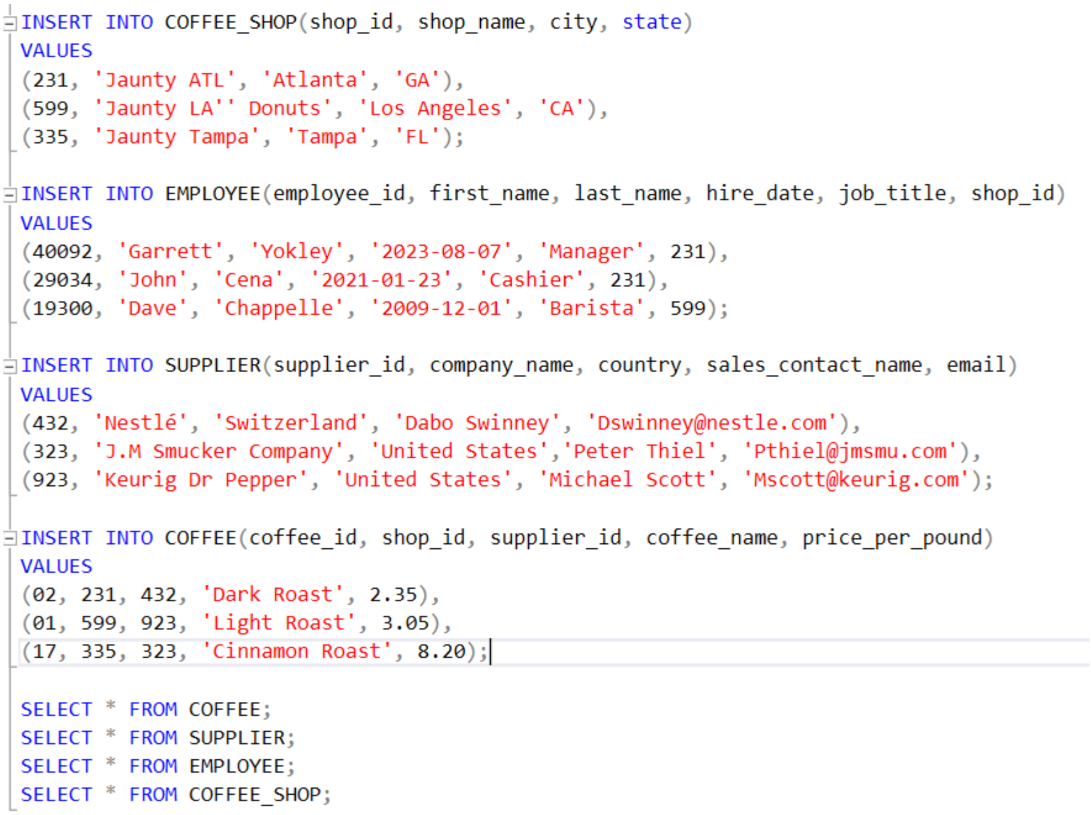
```
CREATE TABLE COFFEE_SHOP(
shop_id INTEGER,
shop_name VARCHAR(50),
city VARCHAR(50),
state CHAR(2),
PRIMARY KEY(shop_id),
);

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)
);

CREATE TABLE EMPLOYEE(
employee_id INTEGER,
first_name VARCHAR(30),
last_name VARCHAR(30),
hire_date DATE,
job_title VARCHAR(30),
shop_id INTEGER,
PRIMARY KEY(employee_id),
FOREIGN KEY(shop_id) REFERENCES COFFEE_SHOP(shop_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),
);
```

```
Commands completed successfully.

Completion time: 2023-08-07T21:55:59.3308527-04:00
```

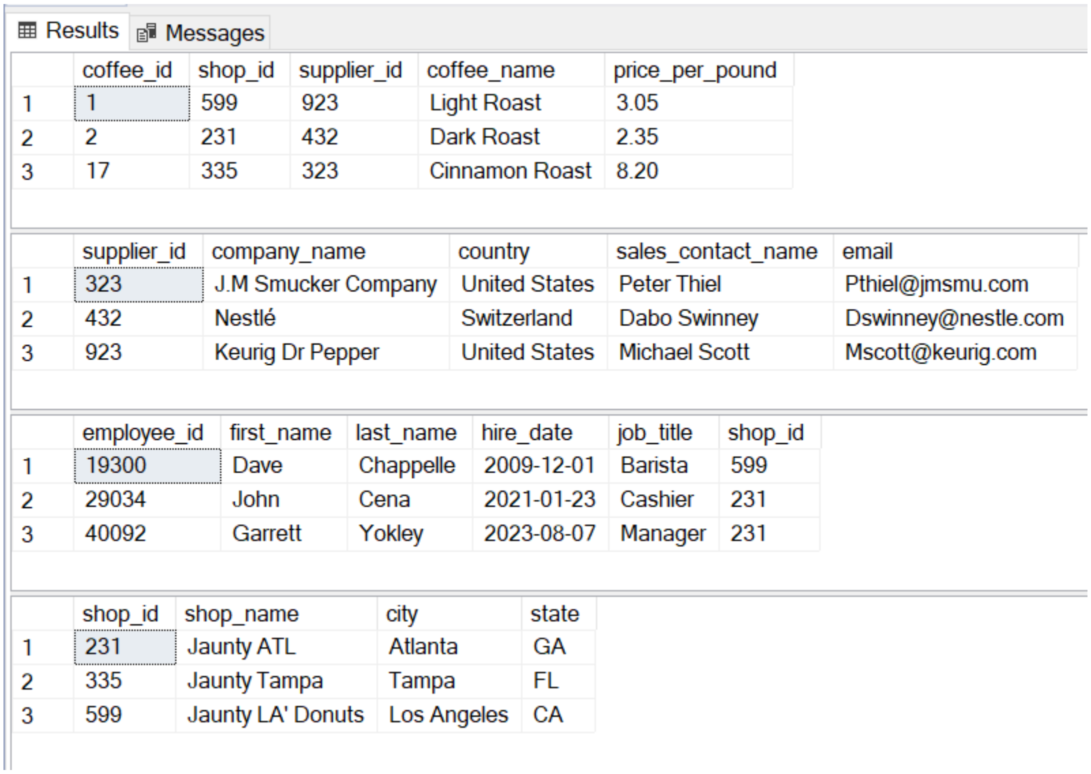


```
⊟ jaunty-coffee
    ⊞ Database Diagrams
    ⊟ Tables
        ⊞ System Tables
        ⊞ FileTables
        ⊞ External Tables
        ⊞ Graph Tables
        ⊟ dbo.COFFEE
            ⊟ Columns
                coffee_id (PK, int, not null)
                shop_id (FK, int, null)
                supplier_id (FK, int, null)
                coffee_name (varchar(30), null)
                price_per_pound (numeric(5,2), null)
            ⊟ Keys
                PK__COFFEE__FE8F721DDEA651E4
                FK__COFFEE__shop_id__3E52440B
                FK__COFFEE__supplier__3F466844
            ⊞ Constraints
            ⊞ Triggers
            ⊞ Indexes
            ⊞ Statistics
        ⊟ dbo.COFFEE_SHOP
            ⊟ Columns
                shop_id (PK, int, not null)
                shop_name (varchar(50), null)
                city (varchar(50), null)
                state (char(2), null)
            ⊟ Keys
                PK__COFFEE_S__AD081786DD3A8D1A
            ⊞ Constraints
            ⊞ Triggers
            ⊞ Indexes
            ⊞ Statistics
        ⊟ dbo.EMPLOYEE
            ⊟ Columns
                employee_id (PK, int, not null)
                first_name (varchar(30), null)
                last_name (varchar(30), null)
                hire_date (date, null)
                job_title (varchar(30), null)
                shop_id (FK, int, null)
            ⊟ Keys
                PK__EMPLOYEE__C52E0BA8202E14DC
                FK__EMPLOYEE__shop_i__3B75D760
            ⊞ Constraints
            ⊞ Triggers
            ⊞ Indexes
            ⊞ Statistics
        ⊟ dbo.SUPPLIER
            ⊟ Columns
                supplier_id (PK, int, not null)
                company_name (varchar(50), null)
                country (varchar(30), null)
                sales_contact_name (varchar(60), null)
                email (varchar(50), not null)
            ⊟ Keys
                PK__SUPPLIER__6EE594E8EC2163E0
            ⊞ Constraints
            ⊞ Triggers
            ⊞ Indexes
            ⊞ Statistics
```

B.2



```
INSERT INTO COFFEE_SHOP(shop_id, shop_name, city, state)
VALUES
(231, 'Jaunty ATL', 'Atlanta', 'GA'),
(599, 'Jaunty LA'' Donuts', 'Los Angeles', 'CA'),
(335, 'Jaunty Tampa', 'Tampa', 'FL');

INSERT INTO EMPLOYEE(employee_id, first_name, last_name, hire_date, job_title, shop_id)
VALUES
(40092, 'Garrett', 'Yokley', '2023-08-07', 'Manager', 231),
(29034, 'John', 'Cena', '2021-01-23', 'Cashier', 231),
(19300, 'Dave', 'Chappelle', '2009-12-01', 'Barista', 599);

INSERT INTO SUPPLIER(supplier_id, company_name, country, sales_contact_name, email)
VALUES
(432, 'Nestlé', 'Switzerland', 'Dabo Swinney', 'Dswinney@nestle.com'),
(323, 'J.M Smucker Company', 'United States','Peter Thiel', 'Pthiel@jmsmu.com'),
(923, 'Keurig Dr Pepper', 'United States', 'Michael Scott', 'Mscott@keurig.com');

INSERT INTO COFFEE(coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
VALUES
(02, 231, 432, 'Dark Roast', 2.35),
(01, 599, 923, 'Light Roast', 3.05),
(17, 335, 323, 'Cinnamon Roast', 8.20);

SELECT * FROM COFFEE;
SELECT * FROM SUPPLIER;
SELECT * FROM EMPLOYEE;
SELECT * FROM COFFEE_SHOP;
```



Results | Messages

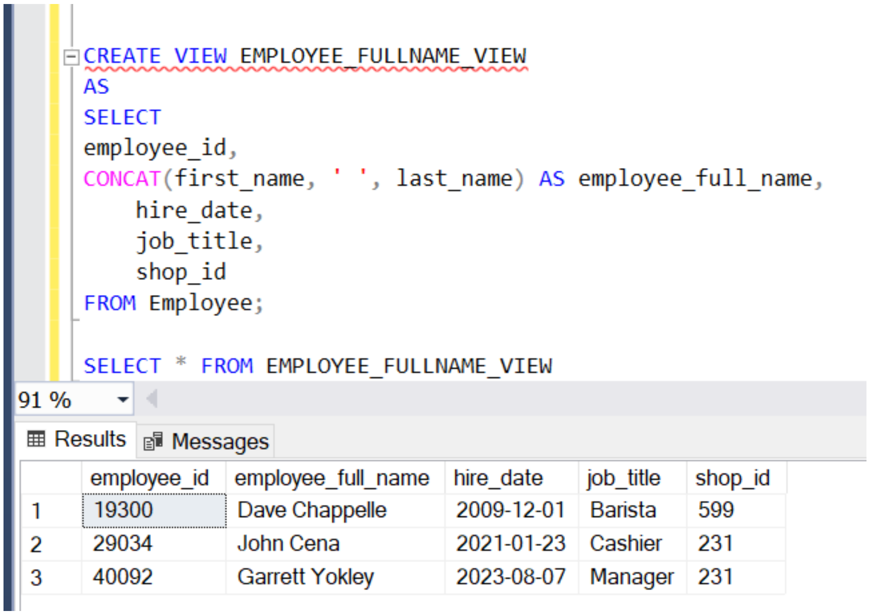
	coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
1	1	599	923	Light Roast	3.05
2	2	231	432	Dark Roast	2.35
3	17	335	323	Cinnamon Roast	8.20

	supplier_id	company_name	country	sales_contact_name	email
1	323	J.M Smucker Company	United States	Peter Thiel	Pthiel@jmsmu.com
2	432	Nestlé	Switzerland	Dabo Swinney	Dswinney@nestle.com
3	923	Keurig Dr Pepper	United States	Michael Scott	Mscott@keurig.com

	employee_id	first_name	last_name	hire_date	job_title	shop_id
1	19300	Dave	Chappelle	2009-12-01	Barista	599
2	29034	John	Cena	2021-01-23	Cashier	231
3	40092	Garrett	Yokley	2023-08-07	Manager	231

	shop_id	shop_name	city	state
1	231	Jaunty ATL	Atlanta	GA
2	335	Jaunty Tampa	Tampa	FL
3	599	Jaunty LA' Donuts	Los Angeles	CA

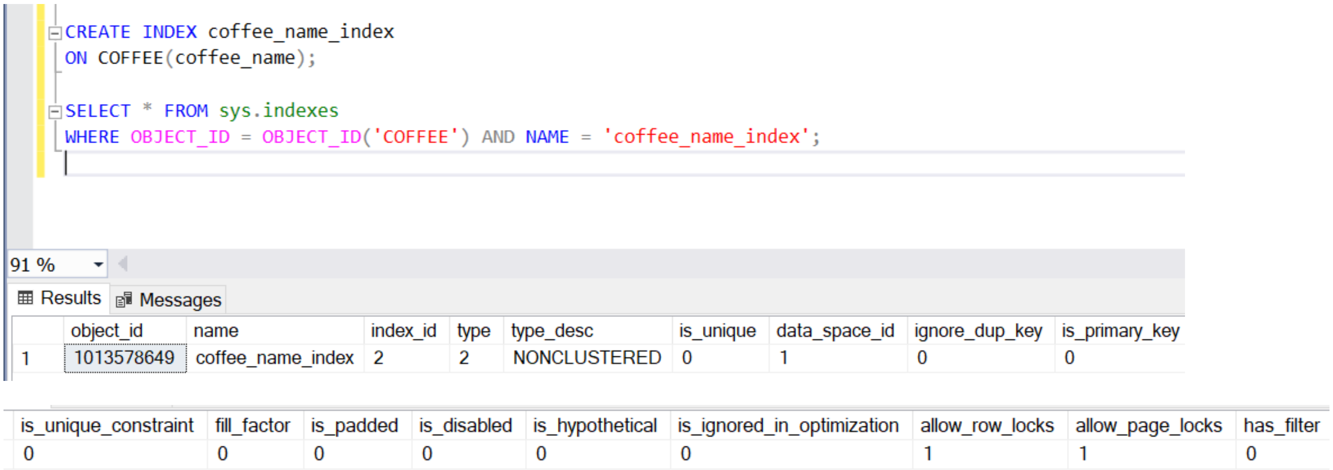
B.3



```
CREATE VIEW EMPLOYEE_FULLNAME_VIEW
AS
SELECT
employee_id,
CONCAT(first_name, ' ', last_name) AS employee_full_name,
    hire_date,
    job_title,
    shop_id
FROM Employee;

SELECT * FROM EMPLOYEE_FULLNAME_VIEW
```

	employee_id	employee_full_name	hire_date	job_title	shop_id
1	19300	Dave Chappelle	2009-12-01	Barista	599
2	29034	John Cena	2021-01-23	Cashier	231
3	40092	Garrett Yokley	2023-08-07	Manager	231

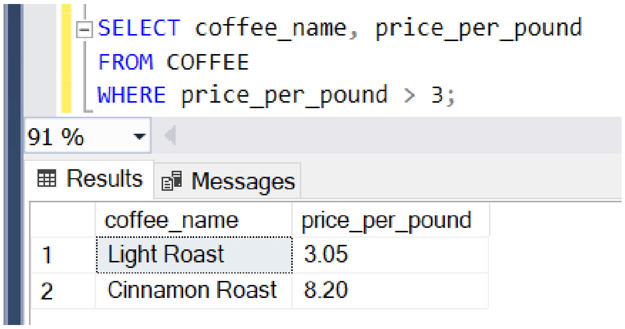
B.4



```sql
CREATE INDEX coffee_name_index
ON COFFEE(coffee_name);

SELECT * FROM sys.indexes
WHERE OBJECT_ID = OBJECT_ID('COFFEE') AND NAME = 'coffee_name_index';
```

	object_id	name	index_id	type	type_desc	is_unique	data_space_id	ignore_dup_key	is_primary_key	is_unique_constraint	fill_factor	is_padded	is_disabled	is_hypothetical	is_ignored_in_optimization	allow_row_locks	allow_page_locks	has_filter
1	1013578649	coffee_name_index	2	2	NONCLUSTERED	0	1	0	0	0	0	0	0	0	0	1	1	0

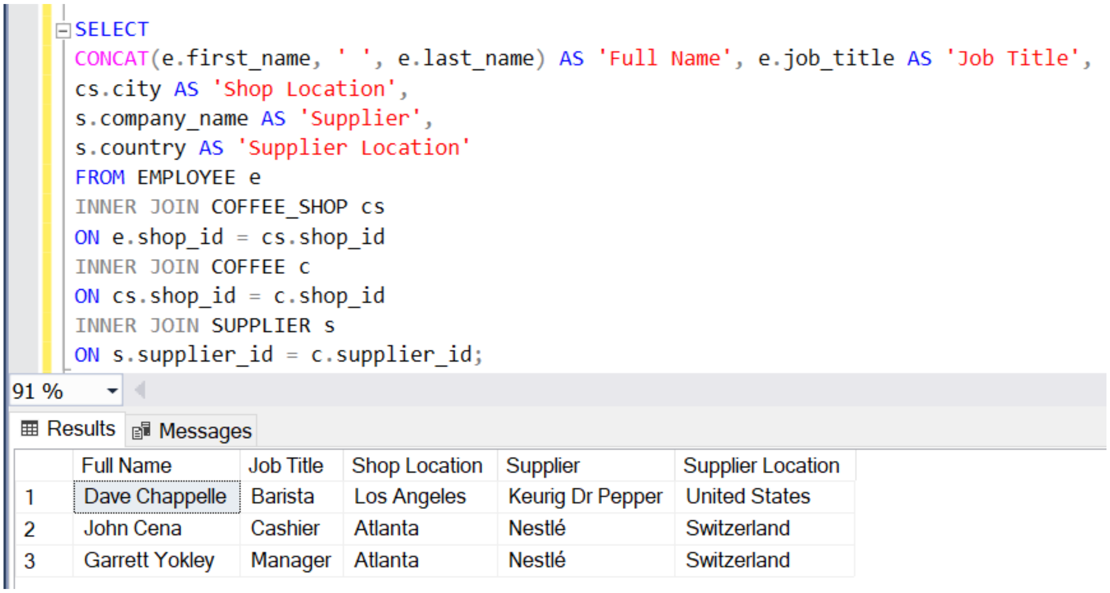
B.5



```sql
SELECT coffee_name, price_per_pound
FROM COFFEE
WHERE price_per_pound > 3;
```

```
     coffee_name      price_per_pound
1    Light Roast      3.05
2    Cinnamon Roast   8.20
```

B.6



```
SELECT
CONCAT(e.first_name, ' ', e.last_name) AS 'Full Name', e.job_title AS 'Job Title',
cs.city AS 'Shop Location',
s.company_name AS 'Supplier',
s.country AS 'Supplier Location'
FROM EMPLOYEE e
INNER JOIN COFFEE_SHOP cs
ON e.shop_id = cs.shop_id
INNER JOIN COFFEE c
ON cs.shop_id = c.shop_id
INNER JOIN SUPPLIER s
ON s.supplier_id = c.supplier_id;
```

	Full Name	Job Title	Shop Location	Supplier	Supplier Location
1	Dave Chappelle	Barista	Los Angeles	Keurig Dr Pepper	United States
2	John Cena	Cashier	Atlanta	Nestlé	Switzerland
3	Garrett Yokley	Manager	Atlanta	Nestlé	Switzerland