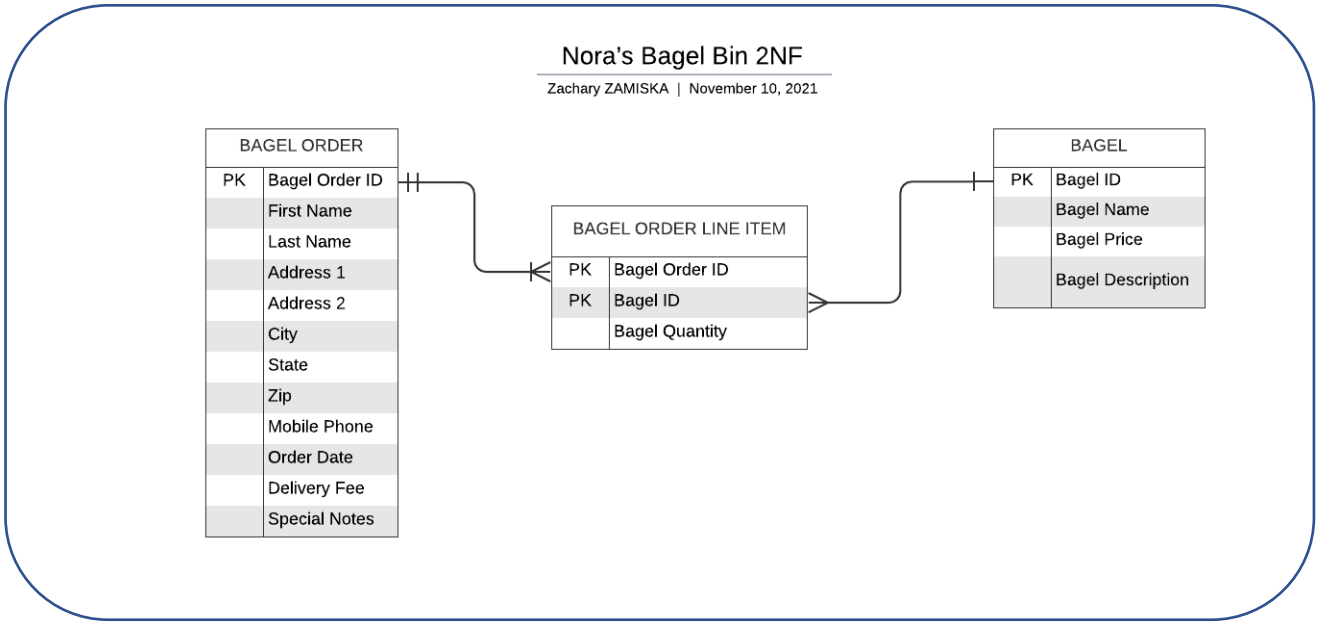
Nora’s Bagel Bin Project

Database Management - Applications

PART A

Zachary Zamiska



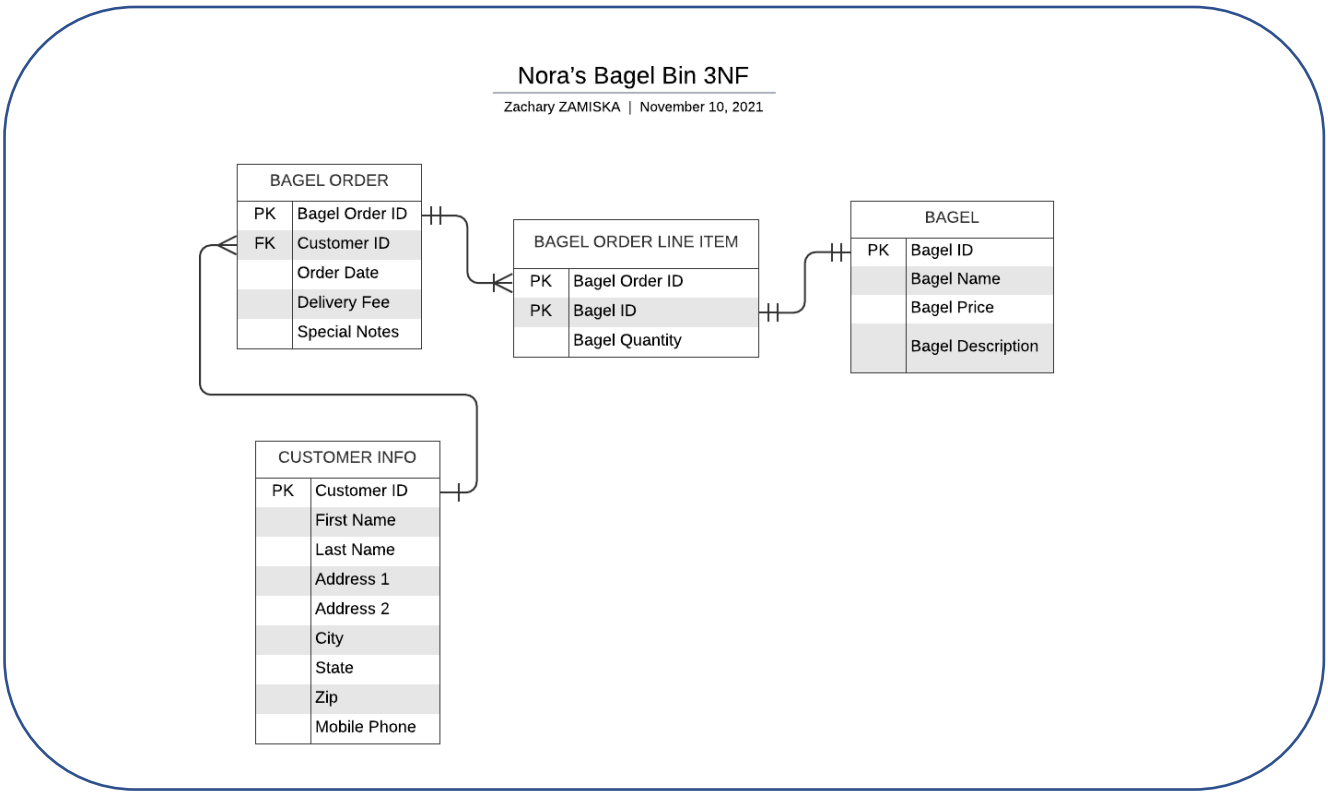
A.1.b

BAGEL ORDER TABLE	--	one-to-many relationship	--	BAGEL ORDER LINE ITEM
BAGEL ORDER LINE ITEM TABLE	--	many-to-one relationship	--	BAGEL TABLE

A.1.c

- The Bagel Order table has a one-to-many relationship to the Bagel Order Line Item table. The reasoning behind this being that there could only be one order per row, however, within that singular order there can be multiple different line items (from the Bagel Order Line Item table).

- In the case of the relationship between the Bagel Order Line Item Table and the Bagel Table, there can be several different line items in several different orders, but only one type of bagel per order line item.



A.2.c

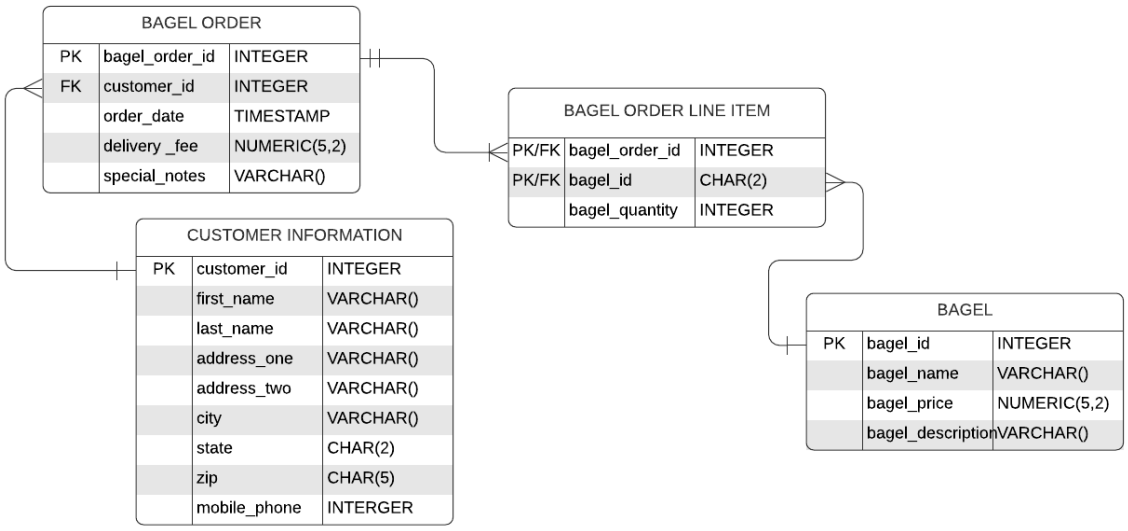
BAGEL ORDER TABLE	many-to-many relationship	CUSTOMER INFO TABLE
BAGEL ORDER TABLE	one-to-many relationship	BAGEL ORDER LINE ITEM TABLE
BAGEL ORDER LINE ITEM TABLE	one-to-one relationship	BAGEL TABLE

A.2.e

- The Bagel Order Table has a one-to-many relationship to the Customer Info Table because a customer can have many different orders containing different things and have many orders in total that can differentiate each time.
- As with the 2NF diagram above, the Bagel Order table has a one-to-many relationship to the Bagel Order Line Item table. The reasoning behind this being that there could only be one order per row, however, within that singular order there can be multiple different line items (from the Bagel Order Line Item table).
- As with the 2NF diagram above, in the case of the relationship between the Bagel Order Line Item Table and the Bagel Table, there can be several different line items in several different orders, but only one type of bagel per order line item.

Nora's Bagel Bin Final Physcial Database Model

Zachary ZAMISKA | November 10, 2021



PART B

Database Management - Applications

Jaunty Coffee Co. ERD

Zachary Zamiska

OBJECTIVES

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* 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 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.

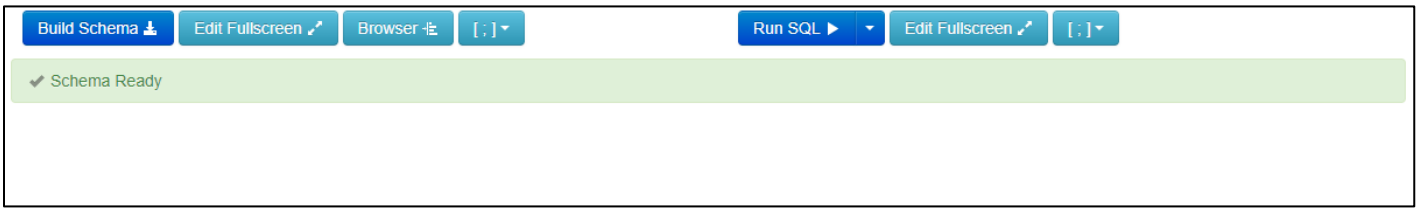
B.1

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

NOTE: All code in this document for part B is compounding.

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

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

CREATE TABLE employee (
  employee_id INTEGER PRIMARY KEY,
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  hire_date DATE,
  job_title VARCHAR(30),
  shop_id INTEGER,

FOREIGN KEY (shop_id) REFERENCES coffee_shop(shop_id)
);

CREATE TABLE coffee (
  coffee_id INTEGER PRIMARY KEY,
  shop_id INTEGER,
  supplier_id INTEGER,
  coffee_name VARCHAR(30),
  price_per_pound NUMERIC(5,2),

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.

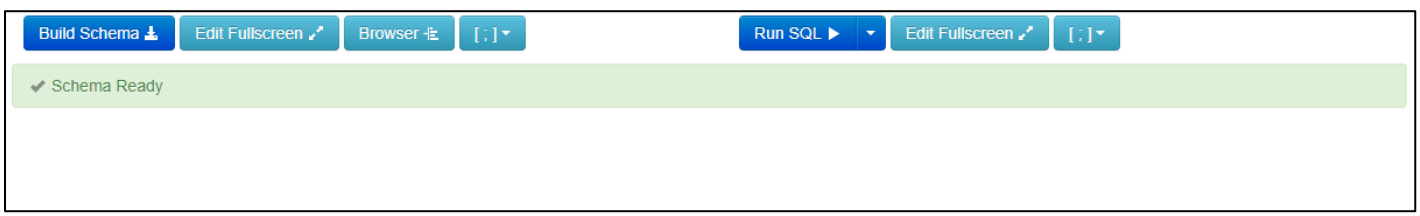


B.2

a. Provide the SQL code you wrote to populate the tables with at least three rows of data in each table.

```sql
INSERT INTO supplier VALUES
  (123, 'La Cafe', 'France', 'Joha Smalt', 'jsmalt@lacafe.org'),
  (124, 'Cals Coffee', 'United States', 'Cal Smith', 'calsmith13@gmail.com'),
  (125, 'Dantino Farms', 'Brazil', 'Jon Galison', 'jgalison@dantinofarms.com'),
  (126, 'Mask Makers Coffee', 'Italy', 'Person Daily', 'personperson@hotmail.com'),
  (127, 'Columbian Forma Farms', 'Norway', 'Hans Zimmer', 'zimmermans@aol.com');

INSERT INTO coffee_shop VALUES
  (1,'Shoppe 1','McDonald','PA'),
  (2,'Shoppe 2','Pittsburgh','PA'),
  (3, 'Shoppe 3', 'Denver','CO'),
  (4, 'Shoppe 4', 'Point Place', 'WI');

INSERT INTO employee VALUES
  (101, 'John', 'Doe', '2020-02-13', 'Assistant Manager', 2),
  (102, 'Eric', 'Foreman', '1978-09-20', 'Team Memeber', 4),
  (103, 'Cole', 'Erican', '2021-01-01', 'Team Member', 3),
  (104, 'Sloane', 'Stevens', '2018-09-02', 'Manager', 4),
  (105, 'Zachary', 'Zamiska', '2019-08-29', 'CEO', 1);

INSERT INTO coffee (coffee_id,shop_id,supplier_id,coffee_name,price_per_pound) VALUES
  (90,2,123,'Carribean',23.23),
  (91,3,127,'American',234.21 ),
  (92,4,125,'Mexican',99.99),
  (93,4,127,'Canadian',142.28);
```

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



B.3

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.

```
SELECT *, CONCAT(first_name , ' ', last_name) AS employee_full_name
FROM employee;
```

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

Build Schema | Edit Fullscreen | Browser | [;] | Run SQL | Edit Fullscreen | [;]

employee_id	first_name	last_name	hire_date	job_title	shop_id	employee_full_name
101	John	Doe	2020-02-13	Assistant Manager	2	John Doe
102	Eric	Foreman	1978-09-20	Team Memeber	4	Eric Foreman
103	Cole	Erican	2021-01-01	Team Member	3	Cole Erican
104	Sloane	Stevens	2018-09-02	Manager	4	Sloane Stevens
105	Zachary	Zamiska	2019-08-29	CEO	1	Zachary Zamiska

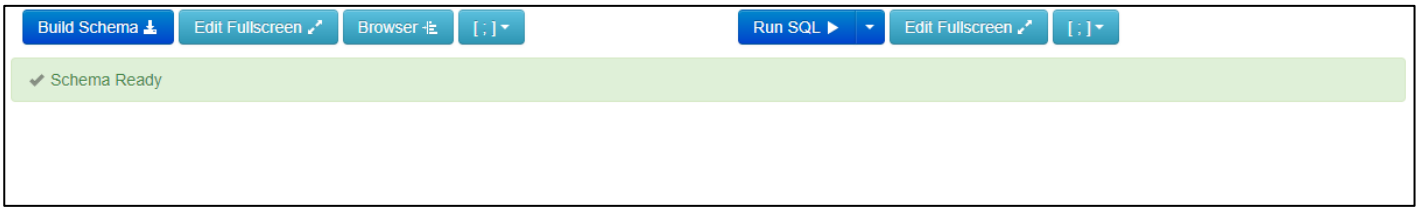
Record Count: 5; Execution Time: 28ms | View Execution Plan | link

B.4

a. Provide the SQL code you wrote to create your index on the coffee_name field from the "Coffee" table.

```
CREATE INDEX index_coffee
ON coffee (coffee_name);

```

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

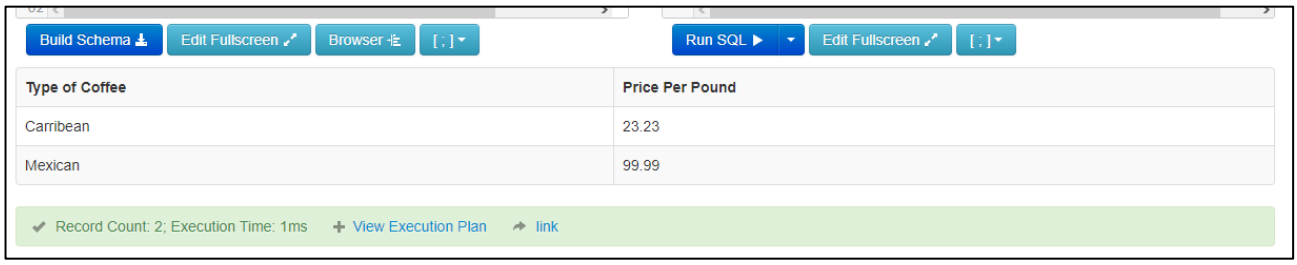


B.5

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

```
SELECT coffee_name AS 'Type of Coffee', price_per_pound AS 'Price Per Pound'
FROM coffee
WHERE coffee.price_per_pound < 100.00;
```

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



B.6

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.

```
SELECT coffee_name AS 'Type of Coffee',company_name AS 'Supplier',shop_name AS 'Shoppe Name'
FROM coffee AS c
   INNER JOIN
   supplier AS s
   ON c.supplier_id = s.supplier_id
   INNER JOIN
   coffee_shop AS cs
   ON c.shop_id = cs.shop_id;
```

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

Build Schema | Edit Fullscreen | Browser | [;] | Run SQL | Edit Fullscreen | [;]

Type of Coffee	Supplier	Shoppe Name
Carribean	La Cafe	Shoppe 2
American	Columbian Forma Farms	Shoppe 3
Mexican	Dantino Farms	Shoppe 4
Canadian	Columbian Forma Farms	Shoppe 4

Record Count: 4; Execution Time: 2ms View Execution Plan link

FINAL CODE

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

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

CREATE TABLE employee (
  employee_id INTEGER PRIMARY KEY,
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  hire_date DATE,
  job_title VARCHAR(30),
  shop_id INTEGER,

FOREIGN KEY (shop_id) REFERENCES coffee_shop(shop_id)
);

CREATE TABLE coffee (
  coffee_id INTEGER PRIMARY KEY,
  shop_id INTEGER,
  supplier_id INTEGER,
  coffee_name VARCHAR(30),
  price_per_pound NUMERIC(5,2),

FOREIGN KEY (shop_id) REFERENCES coffee_shop(shop_id),
FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id)
);

INSERT INTO supplier VALUES
  (123, 'La Cafe', 'France', 'Joha Smalt', 'jsmalt@lacafe.org'),
  (124, 'Cals Coffee', 'United States', 'Cal Smith', 'calsmith13@gmail.com'),
  (125, 'Dantino Farms', 'Brazil', 'Jon Galison', 'jgalison@dantinofarms.com'),
  (126, 'Mask Makers Coffee', 'Italy', 'Person Daily', 'personperson@hotmail.com'),
  (127, 'Columbian Forma Farms', 'Norway', 'Hans Zimmer', 'zimmermans@aol.com');

INSERT INTO coffee_shop VALUES
  (1,'Shoppe 1','McDonald','PA'),
  (2,'Shoppe 2','Pittsburgh','PA'),
  (3, 'Shoppe 3', 'Denver','CO'),
  (4, 'Shoppe 4', 'Point Place', 'WI');

INSERT INTO employee VALUES
  (101, 'John', 'Doe', '2020-02-13', 'Assistant Manager', 2),
  (102, 'Eric', 'Foreman', '1978-09-20', 'Team Memeber', 4),
  (103, 'Cole', 'Erican', '2021-01-01', 'Team Member', 3),
  (104, 'Sloane', 'Stevens', '2018-09-02', 'Manager', 4),
  (105, 'Zachary', 'Zamiska', '2019-08-29', 'CEO', 1);

INSERT INTO coffee (coffee_id,shop_id,supplier_id,coffee_name,price_per_pound) VALUES
  (90,2,123,'Carribean',23.23),
  (91,3,127,'American',234.21 ),
```

```
  (92,4,125,'Mexican',99.99),
  (93,4,127,'Canadian',142.28);

CREATE INDEX index_coffee
ON coffee (coffee_name);



SELECT *, CONCAT(first_name , ' ', last_name) AS employee_full_name
FROM employee;

SELECT coffee_name AS 'Type of Coffee', price_per_pound AS 'Price Per Pound'
FROM coffee
WHERE coffee.price_per_pound < 100.00;


SELECT coffee_name AS 'Type of Coffee',company_name AS 'Supplier',shop_name AS 'Shoppe Name'
FROM coffee AS c
   INNER JOIN
   supplier AS s
   ON c.supplier_id = s.supplier_id
   INNER JOIN
   coffee_shop AS cs
   ON c.shop_id = cs.shop_id;


```