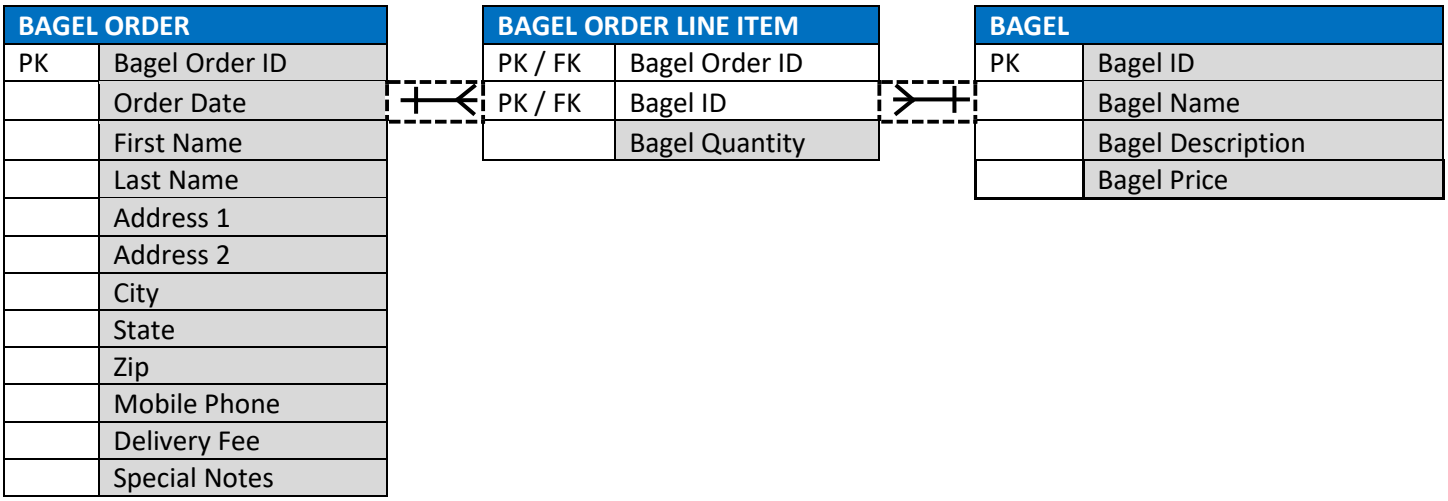
Alireza Barzegar [redacted]

[redacted]

[redacted]

Part A

1.a & b

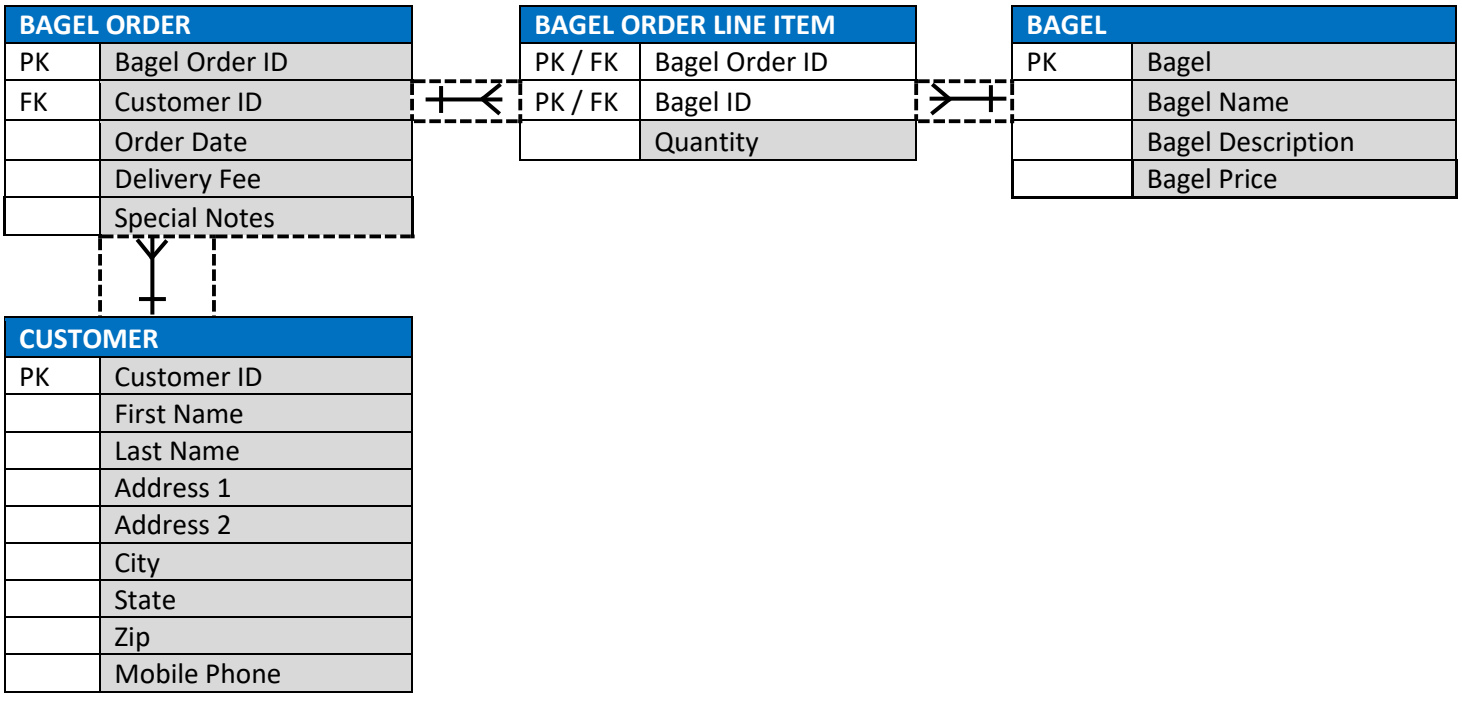


1.c

“Bagel Name”, “Bagel Description”, and “Bagel Price” values are determined by the “Bagel ID” and not the combination of Bagel ID and Bagel Order ID, while the rest of the fields except for “Bagel Quantity” are determined by the Bagel Order ID. So there is a partial functional dependency for these attributes since they depend on part of the primary key and not the whole key. All partial dependencies need to be removed for the tables to reach the 2nd normal form. So “Bagel Name”, “Bagel Description”, and “Bagel Price” need to be moved to their own table called “BAGEL”, where they’ll be fully functionally dependent on the bagel ID as the primary key for the table. The same needs to be done for the remaining fields excluding the “Bagel Quantity”, and in their table Called “BAGEL ORDER” they’ll be fully functionally dependent on the “Bagel Order ID” field. “Bagel Quantity” is fully functionally dependent on the combination of “Bagel Order ID” and “Bagel ID”, so it needs to be moved to its own table, “BAGEL ORDER LINE ITEM”, where it’s fully functionally dependent on the combination (Bagel Order ID, Bagel ID) as its primary key. Also, they’ll each be a foreign key since one is referencing the primary key of the “BAGEL” table and the other is referencing the primary key of the “BAGEL ORDER” table.

Each type of bagel can have a different quantity for different orders, but each quantity is related to one type of bagel only. This can also be seen through the fact that Bagel ID is a primary key in the “BAGEL” table and a foreign key in the “BAGEL ORDER LINE ITEM” table, since a primary key is unique but many instances of a foreign key can reference the same primary key value. Therefore, each instance of the “BAGEL ORDER LINE ITEM” can be related to one instance of the “BAGEL” but each instance of the “BAGEL” can be related to many instances of the “BAGEL ORDER LINE ITEM”. Each order can have multiple types of bagels each with a separate quantity but each quantity of a specific bagel type for a specific order is related to that one order only. Again, This can also be seen through the PK and FK quality that each primary key value is unique but many foreign key instances can reference the same primary key, as Bagel Order ID is a PK in the “BAGEL ORDER” table and a FK in the “Bagel ORDER LINE ITEM”. Therefore, each instance of the “BAGEL ORDER” can be related to many instances of the “BAGEL ORDER LINE ITEM” but each instance of the “BAGEL ORDER LINE ITEM” can be related to one instance of the “BAGEL ORDER”.

2. a, b, c, & d

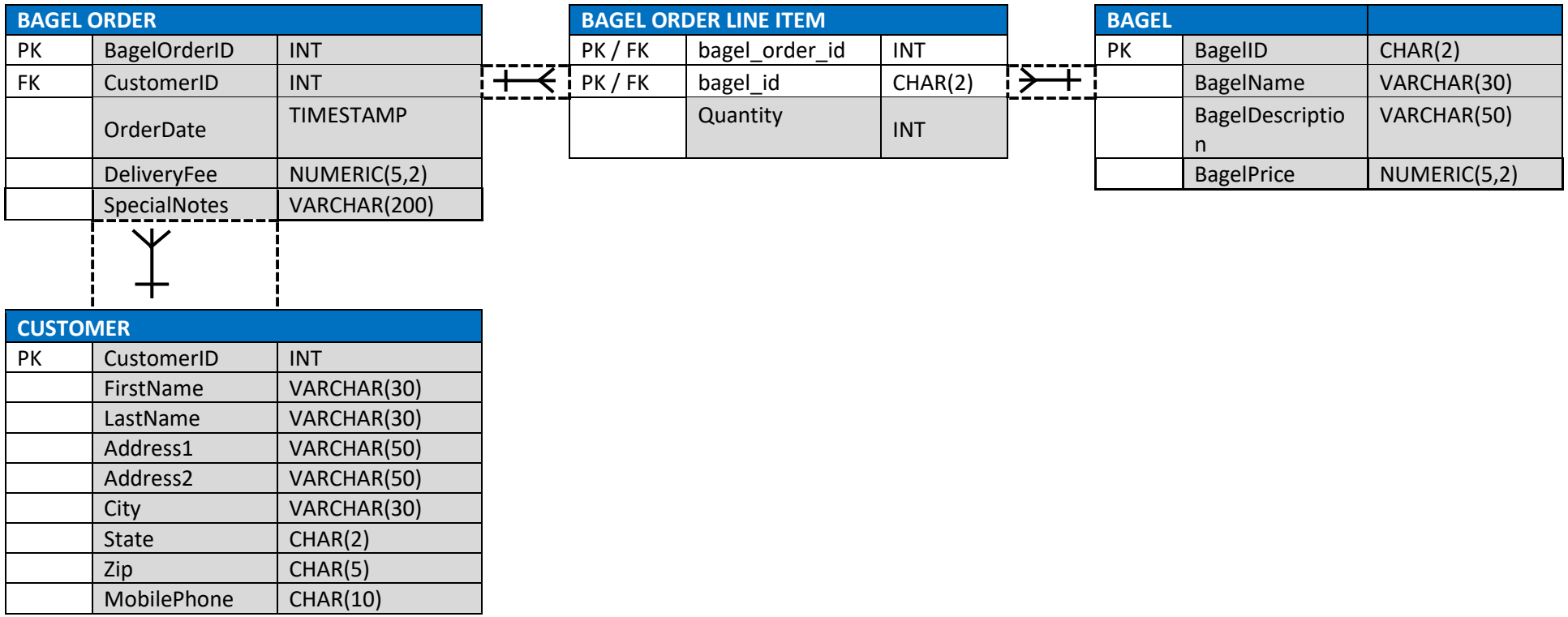


2.e

“Address 1”, “Address 2”, “City”, “State”, “Zip”, and “Mobile Phone” in “BAGEL ORDER” can be determined from the combination of “First Name” and “Last Name”, which is not a key since different customers with the same first name and last name can exist. To reach the third normal form here, all transitive dependencies need to be removed. Therefore, “First Name”, “Last Name”, “Address 1”, “Address 2”, “City”, “State”, “Zip”, and “Mobile Phone” need to be moved from “BAGEL ORDER” to a table where each row will be dependent on an added surrogate key as the primary key. we’ll call this PK “Customer ID”. Since these are all attributes that describe something about the customer, making each row about a specific customer, “CUSTOMER” is a fitting name for the table. To associate each order with the right customer, “customer ID” will become a foreign key in the “BAGEL ORDER” table. No change will be made to the “BAGEL ORDER LINE ITEM” and “BAGEL” tables as they’re already in 3rd normal form with no transitive dependencies.

Each bagel order can be placed by only one customer, but each customer can place many orders. This can also be seen in the fact that “Customer ID” is a foreign key in the “BAGEL ORDER” table and a Primary key in the “CUSTOMER” table because each primary key value is unique while many foreign keys can reference the same primary key value. So, each instance of the “CUSTOMER” table can be related to many instances of the “BAGEL ORDER” table but each instance of the “BAGEL ORDER" table can be related to only one instance of the “customer” table. The remaining cardinalities from A1 remain unchanged.

3.a & b



BAGEL ORDER		
PK	BagelOrderID	INT
FK	CustomerID	INT
	OrderDate	TIMESTAMP
	DeliveryFee	NUMERIC(5,2)
	SpecialNotes	VARCHAR(200)

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

BAGEL		
PK	BagelID	CHAR(2)
	BagelName	VARCHAR(30)
	BagelDescription	VARCHAR(50)
	BagelPrice	NUMERIC(5,2)

CUSTOMER		
PK	CustomerID	INT
	FirstName	VARCHAR(30)
	LastName	VARCHAR(30)
	Address1	VARCHAR(50)
	Address2	VARCHAR(50)
	City	VARCHAR(30)
	State	CHAR(2)
	Zip	CHAR(5)
	MobilePhone	CHAR(10)

Part B

1.a

```
CREATE TABLE  Coffee_Shop (
shop_id  INT  NOT NULL,
shop_name VARCHAR(50),
city VARCHAR(50),
state CHAR(2),
CONSTRAINT  coffeeshopPK  PRIMARY KEY(shop_id)
);

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

CREATE TABLE Coffee (
coffee_id  INT  NOT NULL,
shop_id  INT,
supplier_id  INT,
coffee_name  VARCHAR(30),
price_per_pound  NUMERIC(5,2),
CONSTRAINT  coffeePK  PRIMARY KEY(coffee_id),
CONSTRAINT  shopFKC  FOREIGN KEY(shop_id)  REFERENCES  Coffee_Shop(shop_id),
CONSTRAINT  supplierFKC  FOREIGN KEY(supplier_id)  REFERENCES  Supplier(supplier_id)
);



CREATE TABLE Employee (
```

employee_id INT NOT NULL,

first_name VARCHAR(30),

last_name VARCHAR(30),

hire_date DATE,

job_title VARCHAR(30),

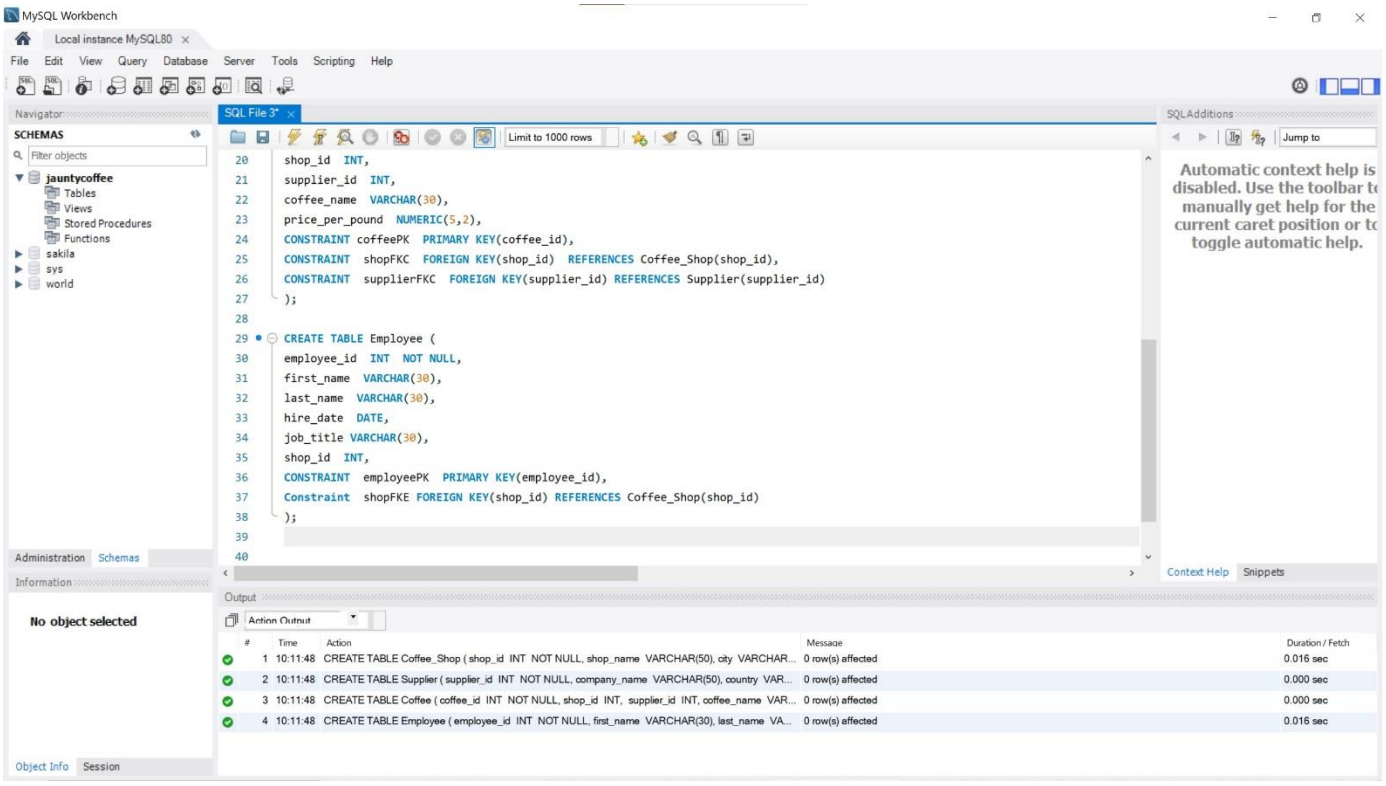
shop_id INT,

CONSTRAINT employeePK PRIMARY KEY(employee_id),

Constraint shopFKE FOREIGN KEY(shop_id) REFERENCES Coffee_Shop(shop_id)

);

1.b



2.a

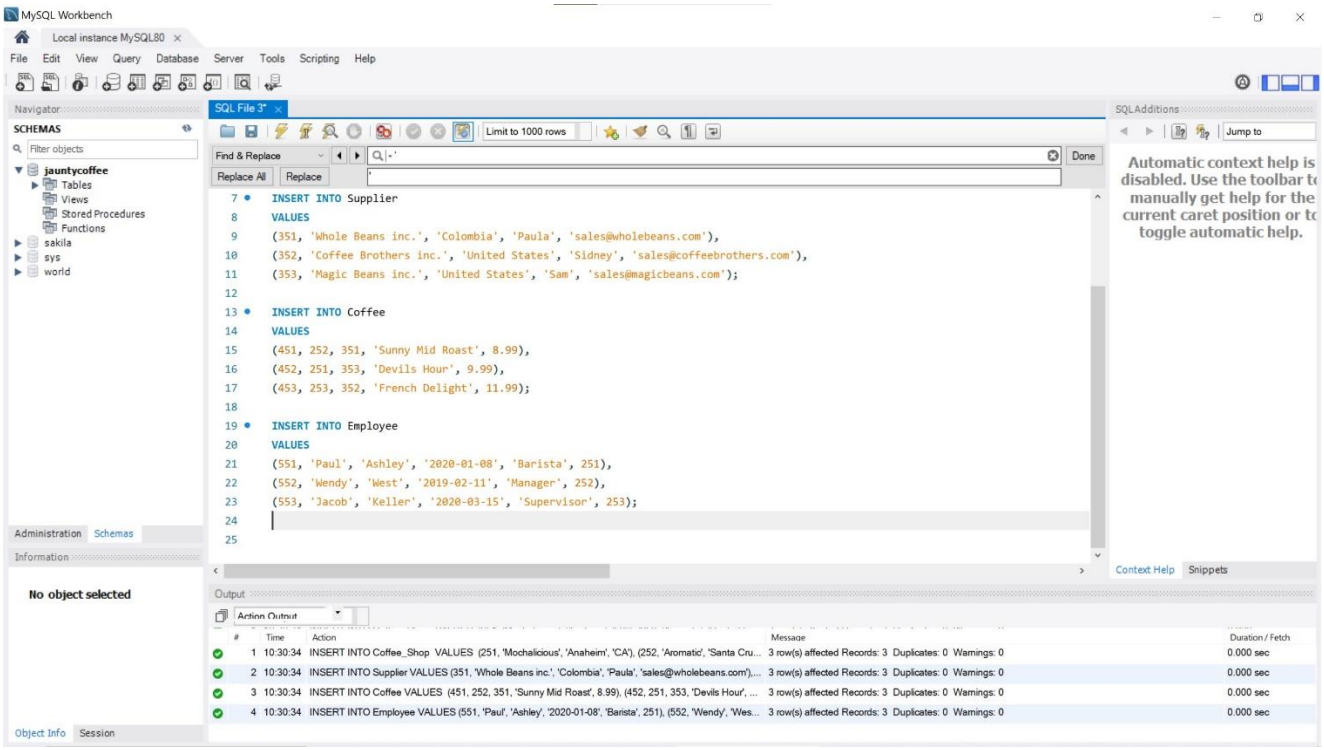
```
INSERT INTO Coffee_Shop
VALUES
(251, 'Mochalicious', 'Anaheim', 'CA'),
(252, 'Aromatic', 'Santa Cruz', 'CA'),
(253, 'BitterSweet', 'Orange', 'CA');

INSERT INTO Supplier
VALUES
(351, 'Whole Beans inc.', 'Colombia', 'Paula', 'sales@wholebeans.com'),
(352, 'Coffee Brothers inc.', 'United States', 'Sidney', 'sales@coffeebrothers.com'),
(353, 'Magic Beans inc.', 'United States', 'Sam', 'sales@magicbeans.com');

INSERT INTO Coffee
VALUES
(451, 252, 351, 'Sunny Mid Roast', 8.99),
(452, 251, 353, 'Devils Hour', 9.99),
(453, 253, 352, 'French Delight', 11.99);

INSERT INTO Employee
VALUES
(551, 'Paul', 'Ashley', '2020-01-08', 'Barista', 251),
(552, 'Wendy', 'West', '2019-02-11', 'Manager', 252),
(553, 'Jacob', 'Keller', '2020-03-15', 'Supervisor', 253);
```

2.b



3.a

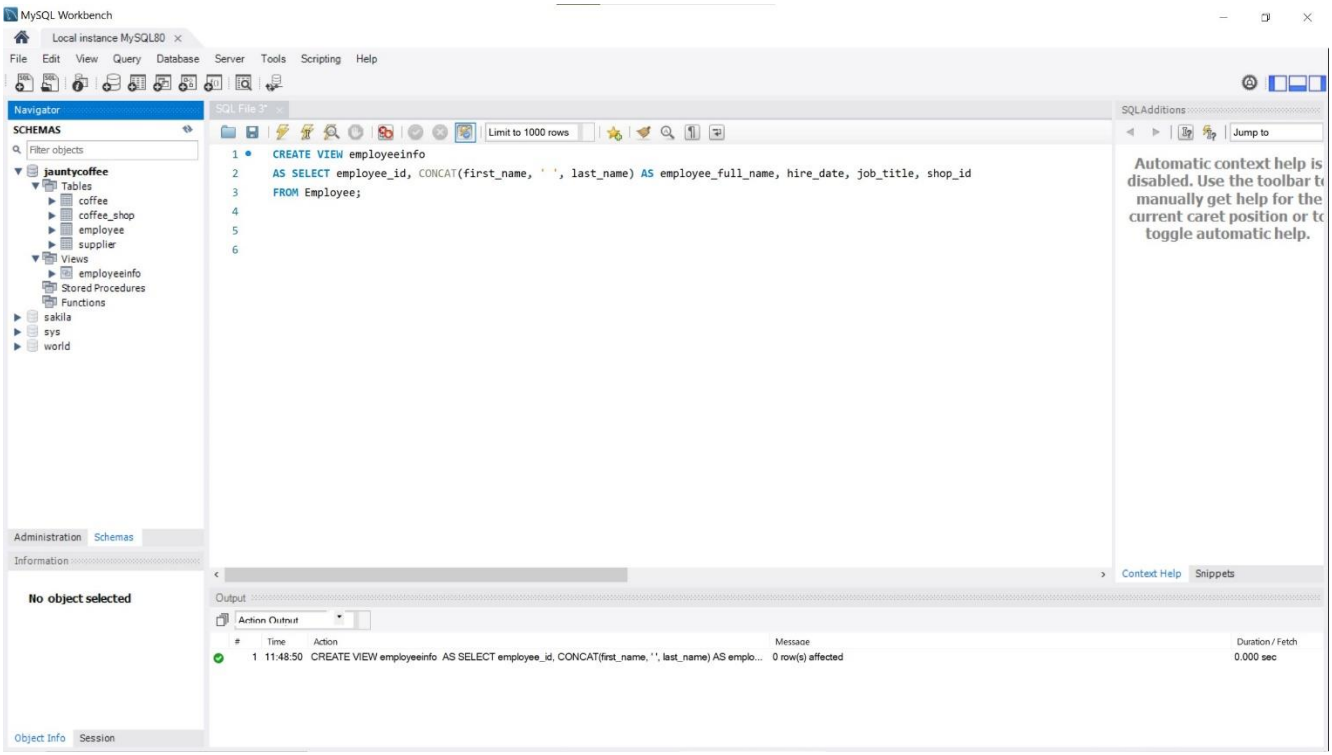
CREATE VIEW employeeinfo

AS SELECT employee_id, CONCAT(first_name, ' ', last_name) AS employee_full_name,

hire_date, job_title, shop_id

FROM Employee;

3.b

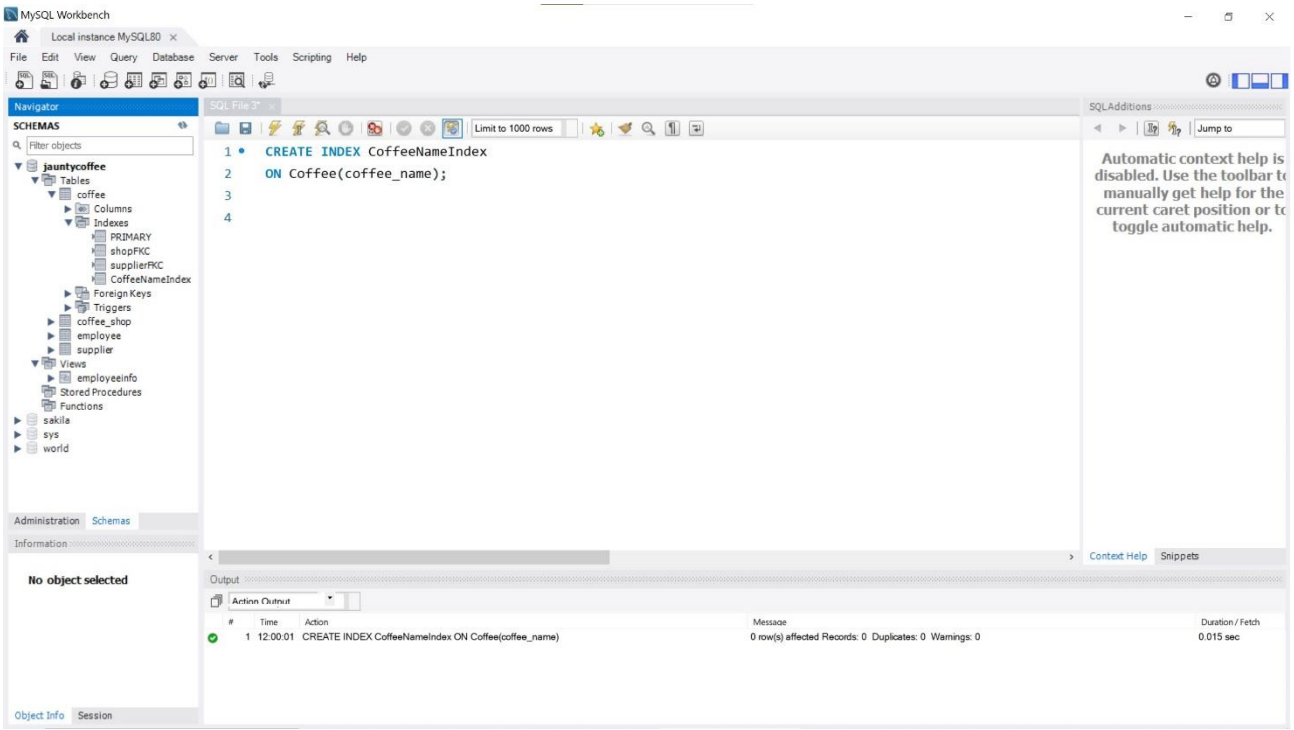


4.a

CREATE INDEX CoffeeNameIndex

ON Coffee(coffee_name);

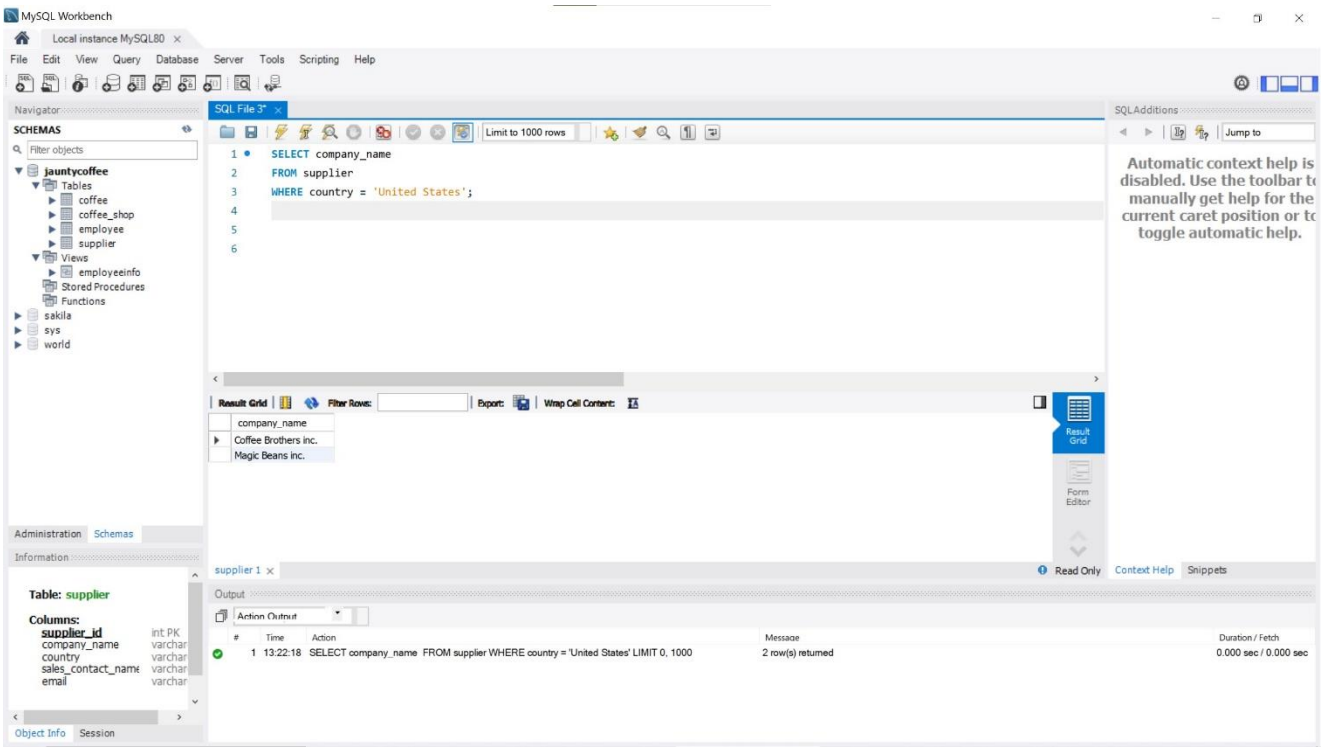
4.b



5.a

```
SELECT  company_name
FROM  supplier
WHERE  country = 'United States';
```

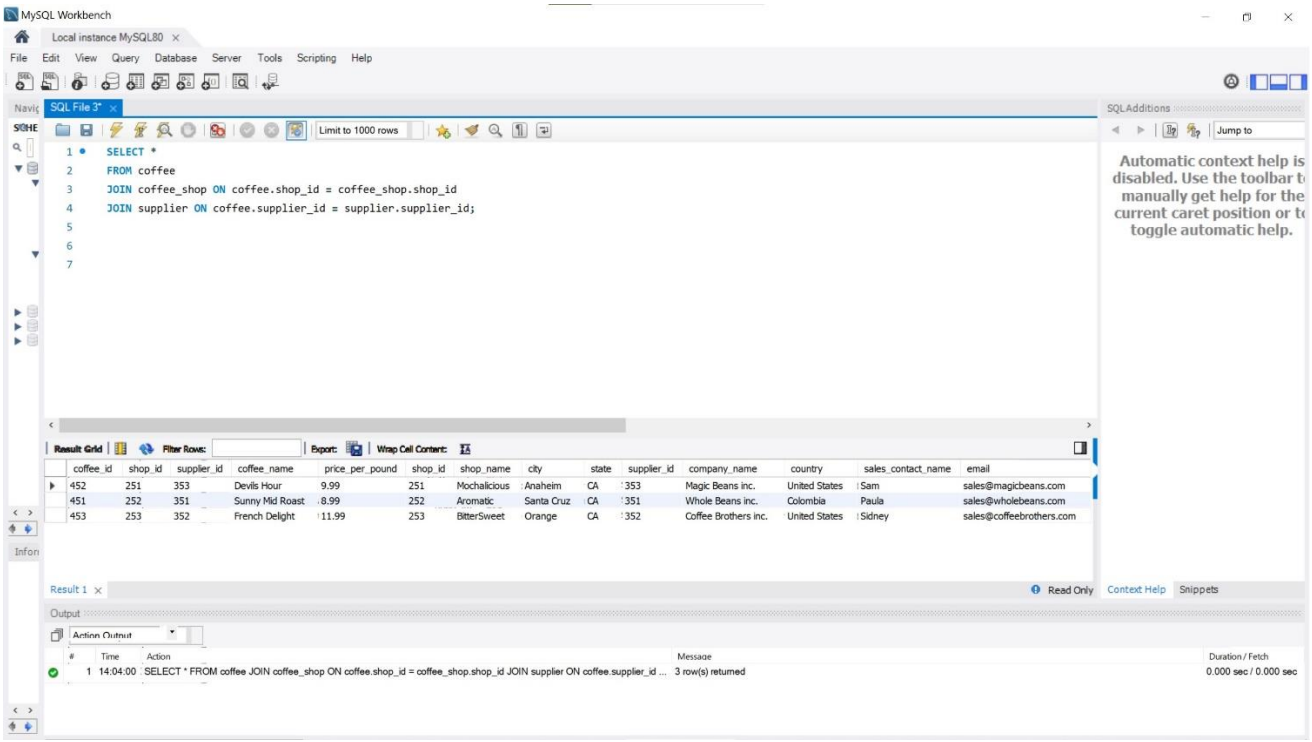
5.b



6.a

```
SELECT *
FROM coffee
JOIN coffee_shop ON coffee.shop_id = coffee_shop.shop_id
JOIN supplier ON coffee.supplier_id = supplier.supplier_id;
```

6.b


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SQL File 3*
Limit to 1000 rows
1 SELECT *
2 FROM coffee
3 JOIN coffee_shop ON coffee.shop_id = coffee_shop.shop_id
4 JOIN supplier ON coffee.supplier_id = supplier.supplier_id;
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
coffee_id shop_id supplier_id coffee_name price_per_pound shop_id shop_name city state supplier_id company_name country sales_contact_name email
452 251 353 Devils Hour 9.99 251 Mochalicious Anaheim CA 353 Magic Beans inc. United States Sam sales@magicbeans.com
451 252 351 Sunny Mid Roast 8.99 252 Aromatic Santa Cruz CA 351 Whole Beans inc. Colombia Paula sales@wholebeans.com
453 253 352 French Delight 11.99 253 BitterSweet Orange CA 352 Coffee Brothers inc. United States Sidney sales@coffeebrothers.com
Result 1
Read Only
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 14:04:00 SELECT * FROM coffee JOIN coffee_shop ON coffee.shop_id = coffee_shop.shop_id JOIN supplier ON coffee.supplier_id ... 3 row(s) returned 0.000 sec / 0.000 sec