Mya Thomas
December 15, 2022
Data Management Application - C170
Student ID: 010507144

Normalization and Database Design

Performance Assessment

Part A: Normalized physical database blueprint for ‘Nora’s Bagel Bin’

Second Normal Form (2NF)

Bagel Order	
PK	**bagel_order_ID**
	order_date
	first_name
	last_name
	address
	city
	state
	zip
	mobile_phone
	delivery_fee
	special_note

(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

1c) Explanation:

After examining the first normal form of the database blueprint for Nora’s Bagel Bin, it was clear that in order to achieve the second normal form I would need to split the attributes into separate tables. To remove partial dependency and reduce redundancy, I moved all non-key attributes into respective tables where they would be fully dependent on the entire primary key. The attributes in the ‘Bagel Order’ table all depend on the ‘bagel_order_ID’ primary key, most of the attributes in this table are related to the customer's information and the order data. In the middle table ‘Bagel Order Line Item’ there are two primary keys along with the ‘bagel_quantity’ attribute. This table will provide information about which order ID belongs to and what quantity and type of bagel. The last table ‘Bagel’ has one primary key labeled ‘bagel_ID’, this table mainly contains data about the name, price, and description of the type of bagel offered at Nora’s Bagel Bin. The 'Bagel Order' table and ‘Bagel Order Line Item’ have a cardinality of one to many (1:M), multiple orders can contain bagels with the same or different ids. Each bagel order item

will contain at most one bagel order. The ‘Bagel’ and ‘Bagel Order Line Item’ tables have a cardinality of (M:1). Each bagel order line item will belong to at most one bagel, and each bagel type can only have one ID, Bagel order line item can be in more than one customers order.

Third Normal Form (3NF)

Customer	
PK	**customer_ID**
	first_name
	last_name
	address
	city
	state
	zip
	mobile_phone

(M:1)

Bagel Order	
PK	**bagel_order_ID**
FK	customer_ID
	order_date
	delivery_fee
	special_note

(1:M)

Bagel Order Line Item	
PK/FK	**bagel_order_ID**
PK/FK	**bagel_ID**
	bagel_quantity

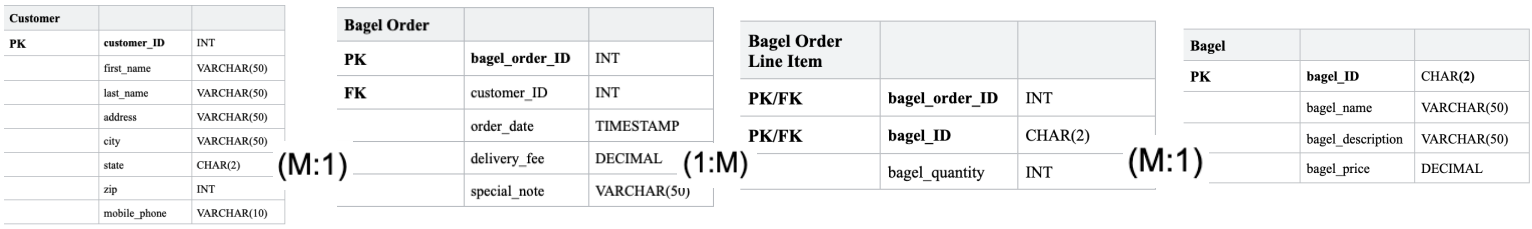
(M:1)

Bagel	
PK	**bagel_ID**
	bagel_name
	bagel_description
	bagel_price

2e) Explanation:

For the table above, I separated the ‘Bagel Order’ table to create the ‘Customer’ table and remove transitive dependencies. Now all the data relating to customers has its own unique table with a primary key named ‘customer_ID’. The bagel order table now has two keys, one primary key named ‘bagel_order_ID’ and one foreign key called ‘customer_ID’ from the customer table. As for the cardinality of the tables above. I chose (M:1) many to-one for the customer and bagel order table. A customer can have multiple orders, each order can have at most one customer.
For the bagel order and bagel order line item table, I chose one to many (1:M). Each bagel order can have many bagel order line items, and at most, each bagel order line item can have one bagel order. Lastly, for the bagel table and bagel order table, I chose many to one (M:1) each bagel order line item will belong to at most one bagel, and each bagel type can only have one ID, Bagel order line item can be in more than one customer order.

Final Physical Database Model



3) Final physical database model, for 'Nora's Bagel Bin' database blueprints.
Every attribute in each table has been assigned a datatype of either CHAR, VARCHAR, TIMESTAMP, INT, or NUMERIC. Each data has been used at least once.

Part B: Develop SQL code to create each table for Jaunty Coffee Co. ERD

1a. SQL code (For table creation):

```
CREATE TABLE `JauntyCoffeeCo.`.`EMPLOYEE`(
        `employee_id` INT NOT NULL,
   `first_name` VARCHAR(30) NOT NULL,
   `last_name` VARCHAR(30) NOT NULL,
   `hire_date` DATE NOT NULL,
   `job_title` VARCHAR(30) NOT NULL,
   `shop_id` INT NOT NULL,
   PRIMARY KEY (`employee_id`),
   FOREIGN KEY (`shop_id`) REFERENCES COFFEE_SHOP(shop_id));

CREATE TABLE `JauntyCoffeeCo.`.`COFFEE_SHOP`(
        `shop_id` INT NOT NULL,
   `shop_name` VARCHAR(50) NOT NULL,
   `city` VARCHAR(50) NOT NULL,
   `state` CHAR(2) NOT NULL,
   PRIMARY KEY (`shop_id`));

CREATE TABLE `JauntyCoffeeCo.`.`COFFEE`(
        `coffee_id` INT NOT NULL,
        `shop_id` INT NOT NULL,
   `supplier_id` INT NOT NULL,
   `coffee_name` VARCHAR(30) NOT NULL,
   `price_per_pound` NUMERIC(5,2) NOT NULL,
```

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

CREATE TABLE `JauntyCoffeeCo.`.`SUPPLIER`(
        `supplier_id` INT NOT NULL,
    `company_name` VARCHAR(50) NOT NULL,
    `country` VARCHAR(30) NOT NULL,
    `sales_contact_name` VARCHAR(60) NOT NULL,
    `email` VARCHAR(50) NOT NULL,
        PRIMARY KEY (`supplier_id`));
```

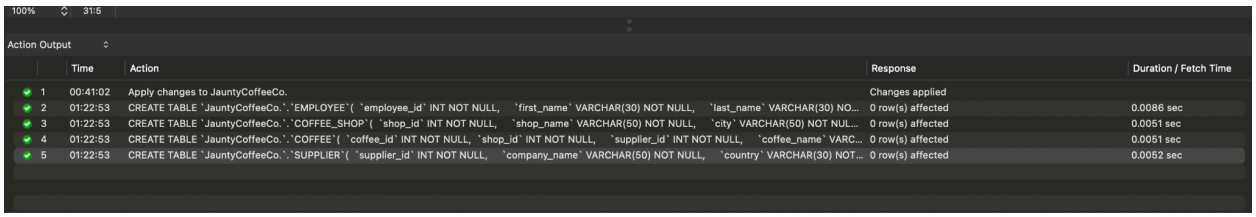
1b. Screenshot of tested code:

```sql
CREATE TABLE `JauntyCoffeeCo.`.`EMPLOYEE`(
    `employee_id` INT NOT NULL,
    `first_name` VARCHAR(30) NOT NULL,
    `last_name` VARCHAR(30) NOT NULL,
    `hire_date` DATE NOT NULL,
    `job_title` VARCHAR(30) NOT NULL,
    `shop_id` INT NOT NULL,
    PRIMARY KEY (`employee_id`),
    FOREIGN KEY (`shop_id`) REFERENCES COFFEE_SHOP(shop_id));

CREATE TABLE `JauntyCoffeeCo.`.`COFFEE_SHOP`(
    `shop_id` INT NOT NULL,
    `shop_name` VARCHAR(50) NOT NULL,
    `city` VARCHAR(50) NOT NULL,
    `state` CHAR(2) NOT NULL,
    PRIMARY KEY (`shop_id`));

CREATE TABLE `JauntyCoffeeCo.`.`COFFEE`(
    `coffee_id` INT NOT NULL,
    `shop_id` INT NOT NULL,
    `supplier_id` INT NOT NULL,
    `coffee_name` VARCHAR(30) NOT NULL,
    `price_per_pound` NUMERIC(5,2) NOT NULL,
    PRIMARY KEY (`coffee_id`),
    FOREIGN KEY (`shop_id`) REFERENCES COFFEE_SHOP(shop_id),
    FOREIGN KEY (`supplier_id`) REFERENCES SUPPLIER(supplier_id));

CREATE TABLE `JauntyCoffeeCo.`.`SUPPLIER`(
    `supplier_id` INT NOT NULL,
    `company_name` VARCHAR(50) NOT NULL,
    `country` VARCHAR(30) NOT NULL,
    `sales_contact_name` VARCHAR(60) NOT NULL,
    `email` VARCHAR(50) NOT NULL,
    PRIMARY KEY (`supplier_id`));
```



100% 31:5

Action Output

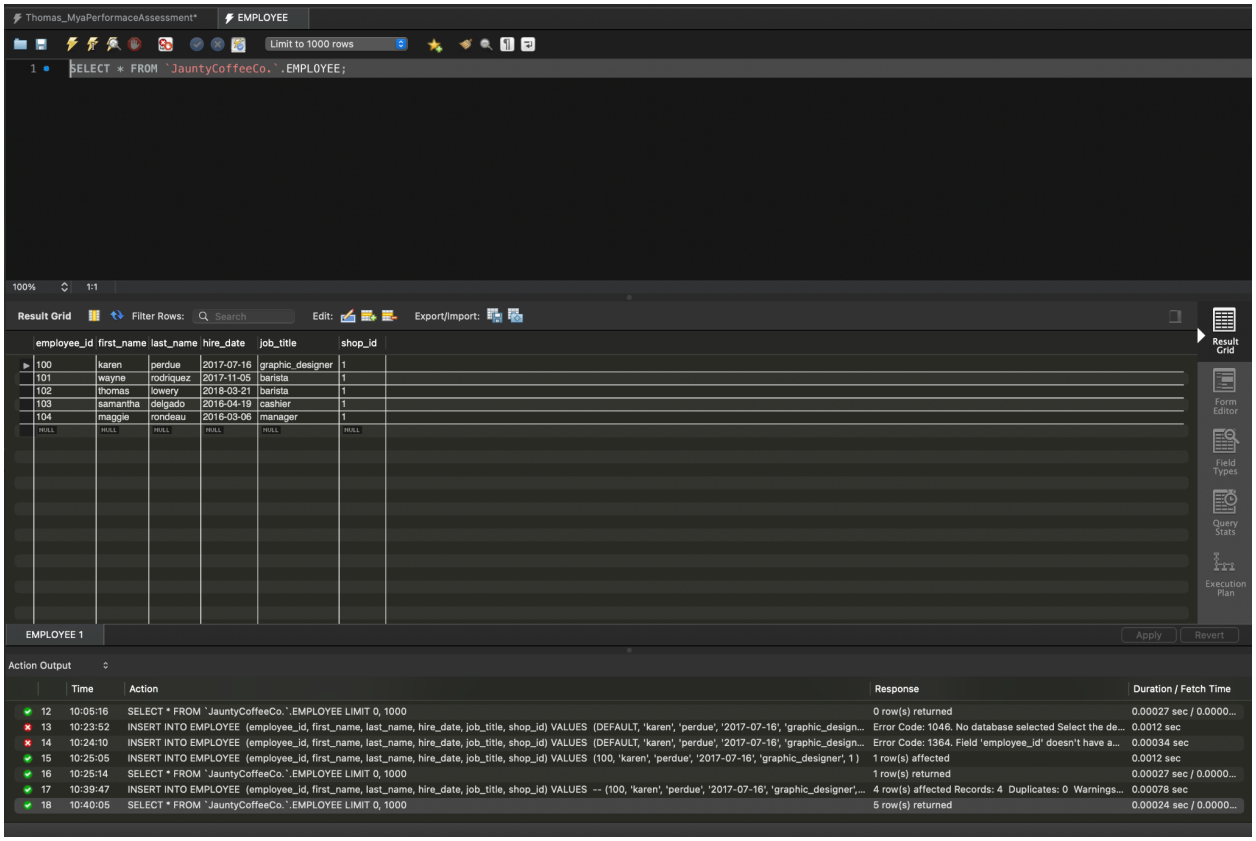
	Time	Action	Response	Duration / Fetch Time
1	00:41:02	Apply changes to JauntyCoffeeCo.	Changes applied	
2	01:22:53	CREATE TABLE `JauntyCoffeeCo.`.`EMPLOYEE`(`employee_id` INT NOT NULL, `first_name` VARCHAR(30) NOT NULL, `last_name` VARCHAR(30) NO...	0 row(s) affected	0.0086 sec
3	01:22:53	CREATE TABLE `JauntyCoffeeCo.`.`COFFEE_SHOP`(`shop_id` INT NOT NULL, `shop_name` VARCHAR(50) NOT NULL, `city` VARCHAR(50) NOT NUL...	0 row(s) affected	0.0051 sec
4	01:22:53	CREATE TABLE `JauntyCoffeeCo.`.`COFFEE`(`coffee_id` INT NOT NULL, `shop_id` INT NOT NULL, `supplier_id` INT NOT NULL, `coffee_name` VARC...	0 row(s) affected	0.0051 sec
5	01:22:53	CREATE TABLE `JauntyCoffeeCo.`.`SUPPLIER`(`supplier_id` INT NOT NULL, `company_name` VARCHAR(50) NOT NULL, `country` VARCHAR(30) NOT...	0 row(s) affected	0.0052 sec

2a.SQL code (For populating the tables with fake data):

****SQL code for entering data into 'EMPLOYEE' table**

```
INSERT INTO EMPLOYEE
        (employee_id, first_name, last_name, hire_date, job_title, shop_id)
VALUES
        (100, 'karen', 'perdue', '2017-07-16', 'graphic_designer', 1 ),
        (101, 'wayne', 'rodriquez', '2017-11-05', 'barista', 1),
        (102, 'thomas', 'lowery', '2018-03-21', 'barista', 1),
        (103, 'samantha', 'delgado', '2016-04-19', 'cashier', 1),
        (104, 'maggie', 'rondeau', '2016-03-06', 'manager', 1);
```

2b. Screenshot of tested code

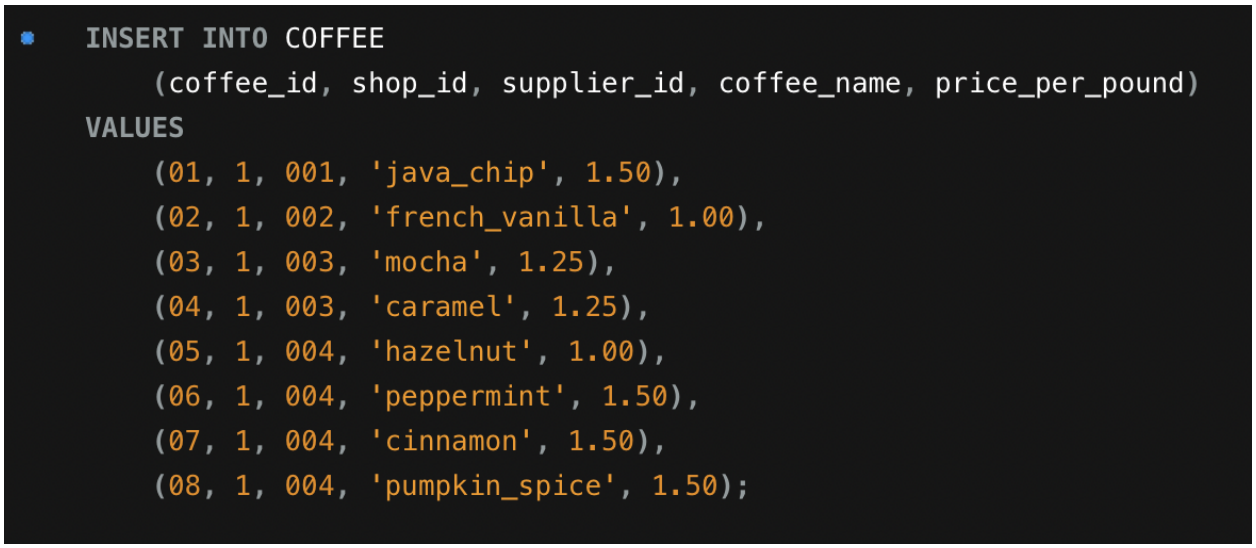



```sql
-- Inserting Data into tables
INSERT INTO EMPLOYEE
    (employee_id, first_name, last_name, hire_date, job_title, shop_id)
VALUES
    (100, 'karen', 'perdue', '2017-07-16', 'graphic_designer', 1 ),
    (101, 'wayne', 'rodriquez', '2017-11-05', 'barista', 1),
    (102, 'thomas', 'lowery', '2018-03-21', 'barista', 1),
    (103, 'samantha', 'delgado', '2016-04-19', 'cashier', 1),
    (104, 'maggie', 'rondeau', '2016-03-06', 'manager', 1);
```

****SQL code for entering data into 'COFFEE' table**

```
INSERT INTO COFFEE
        (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
VALUES
        (01, 1, 001, 'java_chip', 1.50),
        (02, 1, 002, 'french_vanilla', 1.00),
        (03, 1, 003, 'mocha', 1.25),
        (04, 1, 003, 'caramel', 1.25),
        (05, 1, 004, 'hazelnut', 1.00),
        (06, 1, 004, 'peppermint', 1.50),
        (07, 1, 004, 'cinnamon', 1.50),
        (08, 1, 004, 'pumpkin_spice', 1.50);
```

2b. (continued) Screenshot of tested code



```
INSERT INTO COFFEE
    (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
VALUES
    (01, 1, 001, 'java_chip', 1.50),
    (02, 1, 002, 'french_vanilla', 1.00),
    (03, 1, 003, 'mocha', 1.25),
    (04, 1, 003, 'caramel', 1.25),
    (05, 1, 004, 'hazelnut', 1.00),
    (06, 1, 004, 'peppermint', 1.50),
    (07, 1, 004, 'cinnamon', 1.50),
    (08, 1, 004, 'pumpkin_spice', 1.50);
```

****SQL code for entering data into 'SUPPLIER' table**

```
INSERT INTO SUPPLIER
        (supplier_id, company_name, country, sales_contact_name, email)
VALUES
        (001, 'coffee_co.', 'united_states', 'clarence', 'coffee_co@gmail.com'),
        (002, 'rise_&_shine_co.', 'united_states', 'carol', 'rise&shine@gmail.com'),
        (003, 'east_espresso.', 'united_kingdom', 'jeremy', 'eespressoo@gmail.com'),
        (004, 'viva_coco', 'costa_rica', 'jose', 'viva_coco@gmail.com');
```

2b. (continued) Screenshot of tested code

```
INSERT INTO SUPPLIER
    (supplier_id, company_name, country, sales_contact_name, email)
VALUES
    (001, 'coffee_co.', 'united_states', 'clarence', 'coffee_co@gmail.com'),
    (002, 'rise_&_shine_co.', 'united_states', 'carol', 'rise&shine@gmail.com'),
    (003, 'east_espresso.', 'united_kingdom', 'jeremy', 'eespressoo@gmail.com'),
    (004, 'viva_coco', 'costa_rica', 'jose', 'viva_coco@gmail.com');
```

****SQL code for entering data into 'COFFEE_SHOP' table**

```
INSERT INTO COFFEE_SHOP
    (shop_id, shop_name, city, state)
VALUES
    (1, 'jaunty_coffee_co', 'sterling', 'mi');
```

2b. (continued) Screenshot of tested code

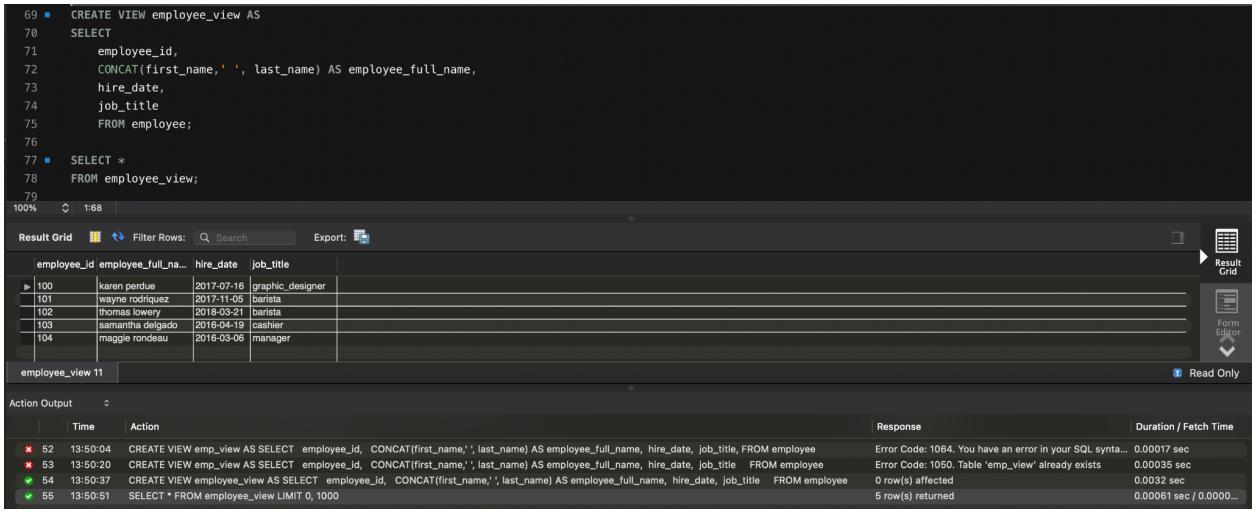
```
INSERT INTO COFFEE_SHOP
    (shop_id, shop_name, city, state)
VALUES
    (1, 'jaunty_coffee_co', 'sterling', 'mi');
```

3a. SQL code (To create a view, containing all information from the employee table. (Concatenate each employee's first and last name, with a space inbetween, then add to new table labeled employee_full_name)):

```
CREATE VIEW employee_view AS
SELECT
    employee_id,
    CONCAT(first_name,' ', last_name) AS employee_full_name,
    hire_date,
    job_title
  FROM employee;

SELECT *
FROM employee_view;
```

3b. Screenshot of tested code

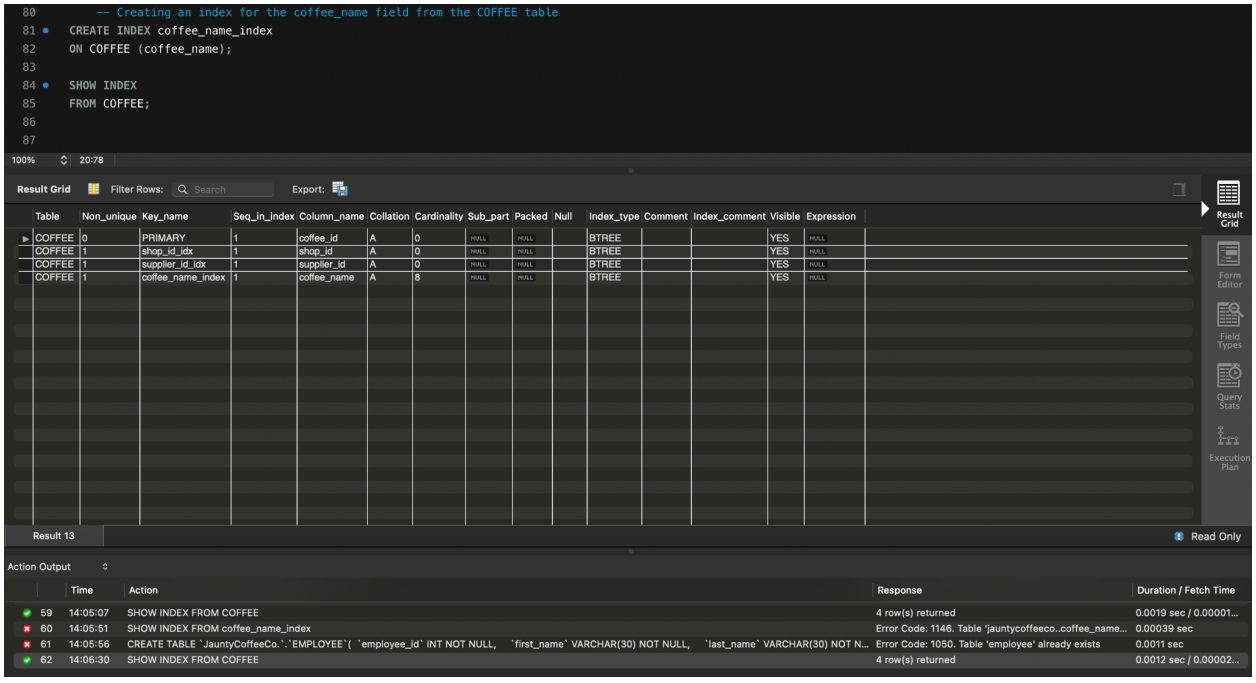


4a. SQL code (Develop code to create an index on the coffee_name field from the Coffee table):

```
-- Creating an index for the coffee_name field from the COFFEE table
CREATE INDEX coffee_name_index
ON COFFEE (coffee_name);

SHOW INDEX
FROM COFFEE;
```

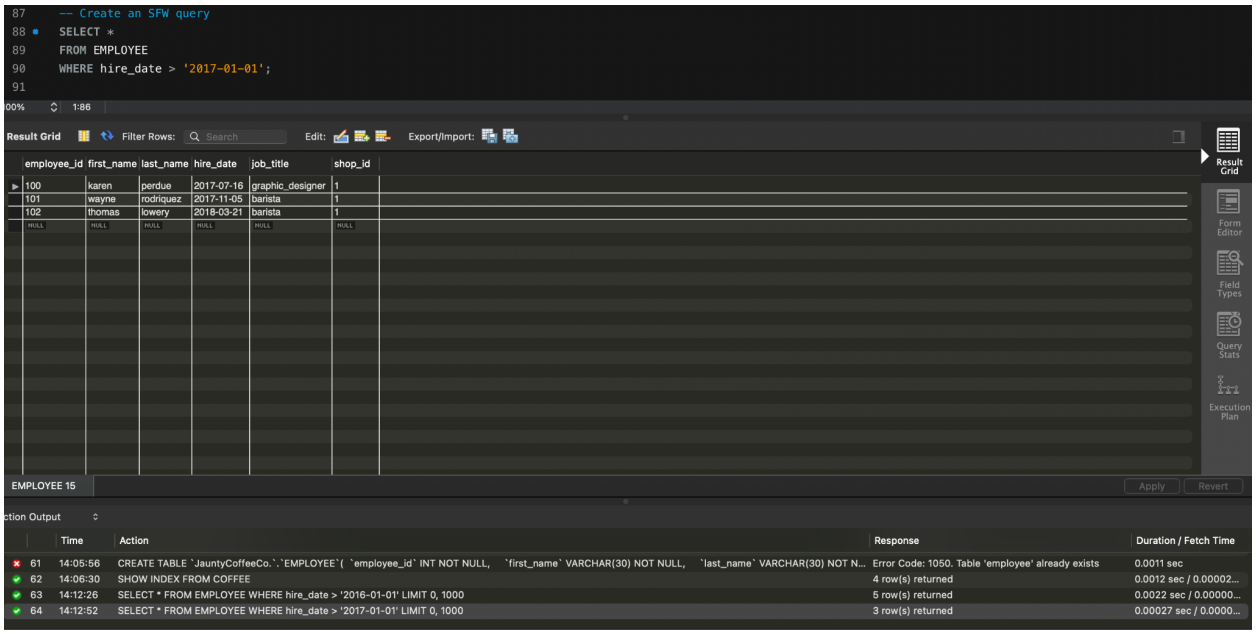
4b. Screenshot of tested code



5a. SQL code (Create a SFW query for any table):

```
-- Create an SFW query
SELECT *
FROM EMPLOYEE
WHERE hire_date > '2017-01-01';
```

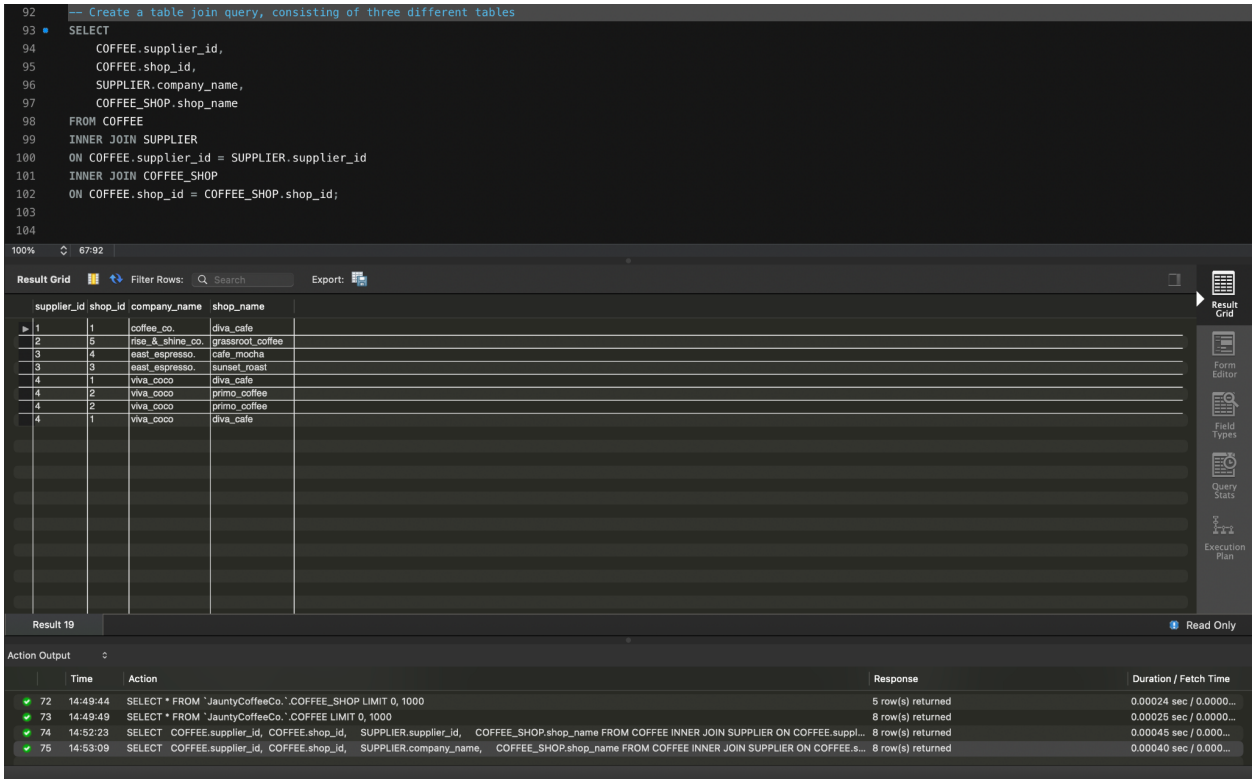
5b. Screenshot of tested code



6a. SQL code (Create a table join query, consisting of three different tables including attributes)

```
-- Create a table join query, consisting of three different tables
SELECT
        COFFEE.supplier_id,
        COFFEE.shop_id,
        SUPPLIER.company_name,
        COFFEE_SHOP.shop_name
FROM COFFEE
INNER JOIN SUPPLIER
ON COFFEE.supplier_id = SUPPLIER.supplier_id
INNER JOIN COFFEE_SHOP
ON COFFEE.shop_id = COFFEE_SHOP.shop_id;
```

6b. Screenshot of tested code



```
92    -- Create a table join query, consisting of three different tables
93    SELECT
94        COFFEE.supplier_id,
95        COFFEE.shop_id,
96        SUPPLIER.company_name,
97        COFFEE_SHOP.shop_name
98    FROM COFFEE
99    INNER JOIN SUPPLIER
100   ON COFFEE.supplier_id = SUPPLIER.supplier_id
101   INNER JOIN COFFEE_SHOP
102   ON COFFEE.shop_id = COFFEE_SHOP.shop_id;
103
104
```

100% 67:92

Result Grid | Filter Rows: Search | Export:

supplier_id	shop_id	company_name	shop_name
1	1	coffee_co.	diva_cafe
2	5	rise_&_shine_co.	grassroot_coffee
3	4	east_espresso.	cafe_mocha
3	3	east_espresso.	sunset_roast
4	1	viva_coco	diva_cafe
4	2	viva_coco	primo_coffee
4	2	viva_coco	primo_coffee
4	1	viva_coco	diva_cafe

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Result 19 — Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
72	14:49:44	SELECT * FROM `JauntyCoffeeCo.`.COFFEE_SHOP LIMIT 0, 1000	5 row(s) returned	0.00024 sec / 0.0000...
73	14:49:49	SELECT * FROM `JauntyCoffeeCo.`.COFFEE LIMIT 0, 1000	8 row(s) returned	0.00025 sec / 0.0000...
74	14:52:23	SELECT COFFEE.supplier_id, COFFEE.shop_id, SUPPLIER.supplier_id, COFFEE_SHOP.shop_name FROM COFFEE INNER JOIN SUPPLIER ON COFFEE.suppl...	8 row(s) returned	0.00045 sec / 0.000...
75	14:53:09	SELECT COFFEE.supplier_id, COFFEE.shop_id, SUPPLIER.company_name, COFFEE_SHOP.shop_name FROM COFFEE INNER JOIN SUPPLIER ON COFFEE.s...	8 row(s) returned	0.00040 sec / 0.000...