

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		Bagel Order		Bagel Order Line Item		Bagel	
PK	customer_ID	PK	bagel_order_ID	PK/FK	bagel_order_ID	PK	bagel_ID
	first_name	FK	customer_ID				bagel_name
	last_name		order_date				bagel_description
	address		delivery_fee	PK/FK	bagel_ID		bagel_price
	city		special_note		bagel_quantity		
	state						
	zip						
	mobile_phone						

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

Customer			Bagel Order			Bagel Order Line Item			Bagel		
PK	customer_ID	INT	PK	bagel_order_ID	INT	PK/FK	bagel_order_ID	INT	PK	bagel_ID	CHAR(2)
	first_name	VARCHAR(50)	FK	customer_ID	INT	PK/FK	bagel_ID	CHAR(2)		bagel_name	VARCHAR(50)
	last_name	VARCHAR(50)		order_date	TIMESTAMP		bagel_quantity	INT		bagel_description	VARCHAR(50)
	address	VARCHAR(50)		delivery_fee	DECIMAL (1:M)					bagel_price	DECIMAL
	city	VARCHAR(50)		special_note	VARCHAR(50)						
	state	CHAR(2)	(M:1)					(M:1)			
	zip	INT									
	mobile_phone	VARCHAR(10)									

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:

```
1 CREATE TABLE `JauntyCoffeeCo`.`EMPLOYEE` (  
2     `employee_id` INT NOT NULL,  
3     `first_name` VARCHAR(30) NOT NULL,  
4     `last_name` VARCHAR(30) NOT NULL,  
5     `hire_date` DATE NOT NULL,  
6     `job_title` VARCHAR(30) NOT NULL,  
7     `shop_id` INT NOT NULL,  
8     PRIMARY KEY (`employee_id`),  
9     FOREIGN KEY (`shop_id`) REFERENCES COFFEE_SHOP(shop_id));  
10  
11 CREATE TABLE `JauntyCoffeeCo`.`COFFEE_SHOP` (  
12     `shop_id` INT NOT NULL,  
13     `shop_name` VARCHAR(50) NOT NULL,  
14     `city` VARCHAR(50) NOT NULL,  
15     `state` CHAR(2) NOT NULL,  
16     PRIMARY KEY (`shop_id`));  
17  
18 CREATE TABLE `JauntyCoffeeCo`.`COFFEE` (  
19     `coffee_id` INT NOT NULL,  
20     `shop_id` INT NOT NULL,  
21     `supplier_id` INT NOT NULL,  
22     `coffee_name` VARCHAR(30) NOT NULL,  
23     `price_per_pound` NUMERIC(5,2) NOT NULL,  
24     PRIMARY KEY (`coffee_id`),  
25     FOREIGN KEY (`shop_id`) REFERENCES COFFEE_SHOP(shop_id),  
26     FOREIGN KEY (`supplier_id`) REFERENCES SUPPLIER(supplier_id));  
27  
28 CREATE TABLE `JauntyCoffeeCo`.`SUPPLIER` (  
29     `supplier_id` INT NOT NULL,  
30     `company_name` VARCHAR(50) NOT NULL,  
31     `country` VARCHAR(30) NOT NULL,  
32     `sales_contact_name` VARCHAR(60) NOT NULL,  
33     `email` VARCHAR(50) NOT NULL,  
34     PRIMARY KEY (`supplier_id`));
```

Action Output				
	Time	Action	Response	Duration / Fetch Time
✓	00:41:02	Apply changes to JauntyCoffeeCo.	Changes applied	
✓	01:22:53	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));	0 row(s) affected	0.0086 sec
✓	01:22:53	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`));	0 row(s) affected	0.0051 sec
✓	01:22:53	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));	0 row(s) affected	0.0051 sec
✓	01:22:53	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`));	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

The screenshot shows a database management tool interface. At the top, there's a tab labeled 'EMPLOYEE'. Below it, a query editor shows the SQL query: `SELECT * FROM 'JauntyCoffeeCo.'.EMPLOYEE;`. The query is executed, and the results are displayed in a table with the following columns: `employee_id`, `first_name`, `last_name`, `hire_date`, `job_title`, and `shop_id`. The table contains five rows of data:

employee_id	first_name	last_name	hire_date	job_title	shop_id
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

Below the table, there's an 'Action Output' section showing the execution of the SQL code. It includes the following actions and their results:

- 12: 10:05:18: `SELECT * FROM 'JauntyCoffeeCo.'.EMPLOYEE LIMIT 0, 1000` - 0 row(s) returned - 0.00027 sec / 0.0000...
- 13: 10:23:52: `INSERT INTO EMPLOYEE (employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES (DEFAULT, 'karen', 'perdue', '2017-07-16', 'graphic_designer', 1)` - Error Code: 1046. No database selected Select the de... - 0.0012 sec
- 14: 10:24:10: `INSERT INTO EMPLOYEE (employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES (DEFAULT, 'karen', 'perdue', '2017-07-16', 'graphic_designer', 1)` - Error Code: 1364. Field 'employee_id' doesn't have a... - 0.00034 sec
- 15: 10:25:05: `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)` - 1 row(s) affected - 0.0012 sec
- 16: 10:25:14: `SELECT * FROM 'JauntyCoffeeCo.'.EMPLOYEE LIMIT 0, 1000` - 1 row(s) returned - 0.00027 sec / 0.0000...
- 17: 10:39:47: `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)` - 4 row(s) affected Records: 4 Duplicates: 0 Warnings... - 0.00078 sec
- 18: 10:40:05: `SELECT * FROM 'JauntyCoffeeCo.'.EMPLOYEE LIMIT 0, 1000` - 5 row(s) returned - 0.00024 sec / 0.0000...

```
-- 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**

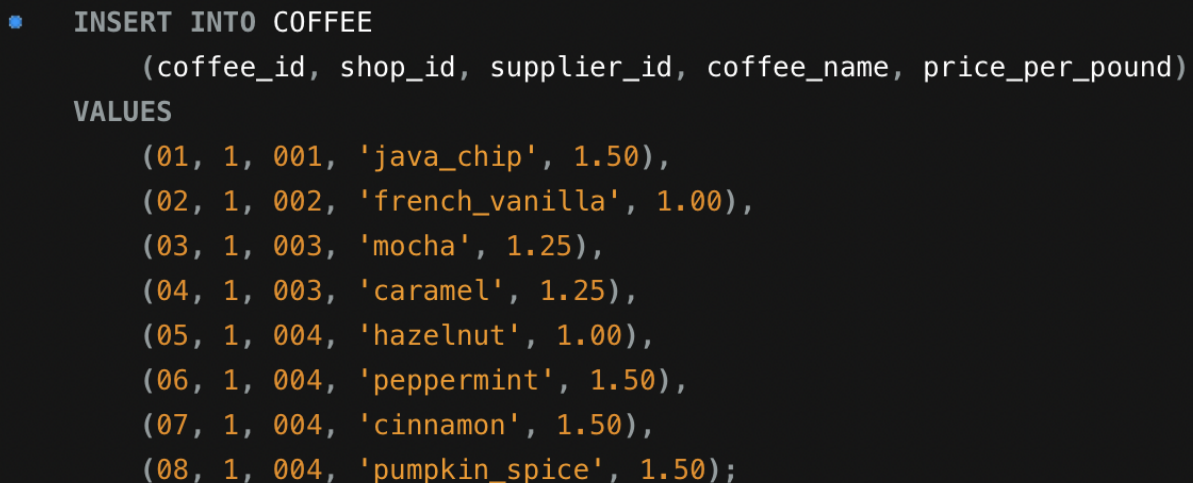
INSERT INTO COFFEE

(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', 'eespresso@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', 'eespresso@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

```
69 CREATE VIEW employee_view AS
70 SELECT
71     employee_id,
72     CONCAT(first_name, ' ', last_name) AS employee_full_name,
73     hire_date,
74     job_title
75 FROM employee;
76
77 SELECT *
78 FROM employee_view;
```

Result Grid

employee_id	employee_full_name	hire_date	job_title
100	karen perdue	2017-07-16	graphic_designer
101	wayne rodriguez	2017-11-05	barista
102	thomas lowery	2018-03-21	barista
103	samantha delgado	2018-04-19	cashier
104	maggie rondau	2018-03-06	manager

employee_view 11

Action Output

	Time	Action	Response	Duration / Fetch Time
✖	52	13:50:04	CREATE VIEW emp_view AS SELECT employee_id, CONCAT(first_name, ' ', last_name) AS employee_full_name, hire_date, job_title, FROM employee	Error Code: 1064. You have an error in your SQL syntax... 0.00017 sec
✖	53	13:50:20	CREATE VIEW emp_view AS SELECT employee_id, CONCAT(first_name, ' ', last_name) AS employee_full_name, hire_date, job_title FROM employee	Error Code: 1050. Table 'emp_view' already exists 0.00035 sec
✔	54	13:50:37	CREATE VIEW employee_view AS SELECT employee_id, CONCAT(first_name, ' ', last_name) AS employee_full_name, hire_date, job_title FROM employee	0 row(s) affected 0.0032 sec
✔	55	13:50:51	SELECT * FROM employee_view LIMIT 0, 1000	5 row(s) returned 0.00061 sec / 0.00000...

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

```
80 -- Creating an index for the coffee_name field from the COFFEE table
81 CREATE INDEX coffee_name_index
82 ON COFFEE (coffee_name);
83
84 SHOW INDEX
85 FROM COFFEE;
```

Result Grid

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
COFFEE	0	PRIMARY	1	coffee_id	A	0				BTREE			YES	
COFFEE	1	shop_id_idx	1	shop_id	A	0				BTREE			YES	
COFFEE	1	supplier_id_idx	1	supplier_id	A	0				BTREE			YES	
COFFEE	1	coffee_name_index	1	coffee_name	A	8				BTREE			YES	

Result 13

Action Output

	Time	Action	Response	Duration / Fetch Time
✔	59	14:05:07	SHOW INDEX FROM COFFEE	4 row(s) returned 0.0019 sec / 0.00001...
✖	60	14:05:51	SHOW INDEX FROM coffee_name_index	Error Code: 1146. Table 'jauntycoffeeco.coffee_name...' 0.00039 sec
✖	61	14:05:56	CREATE TABLE 'JauntyCoffeeCo.' 'EMPLOYEE' ('employee_id' INT NOT NULL, 'first_name' VARCHAR(30) NOT NULL, 'last_name' VARCHAR(30) NOT N...	Error Code: 1050. Table 'employee' already exists 0.0011 sec
✔	62	14:06:30	SHOW INDEX FROM COFFEE	4 row(s) returned 0.0012 sec / 0.00002...

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

The screenshot shows a SQL IDE interface. The top panel displays the SQL query: `-- Create an SFW query`, `SELECT *`, `FROM EMPLOYEE`, `WHERE hire_date > '2017-01-01';`. The middle panel shows the 'Result Grid' with columns: `employee_id`, `first_name`, `last_name`, `hire_date`, `job_title`, and `shop_id`. The data rows are:

employee_id	first_name	last_name	hire_date	job_title	shop_id
100	karen	perdue	2017-07-16	graphic_designer	1
101	wayne	rodriguez	2017-11-09	barista	1
102	thomas	lowery	2018-03-21	barista	1
NULL	NULL	NULL	NULL	NULL	NULL

The bottom panel shows the 'Execution Output' with a table of execution details:

Time	Action	Response	Duration / Fetch Time
61 14:05:56	CREATE TABLE 'JauntyCoffeeCo.', 'EMPLOYEE' ('employee_id' INT NOT NULL, 'first_name' VARCHAR(30) NOT NULL, 'last_name' VARCHAR(30) NOT N...	Error Code: 1050. Table 'employee' already exists	0.0011 sec
62 14:06:30	SHOW INDEX FROM COFFEE	4 row(s) returned	0.0012 sec / 0.00002...
63 14:12:26	SELECT * FROM EMPLOYEE WHERE hire_date > '2016-01-01' LIMIT 0, 1000	5 row(s) returned	0.0022 sec / 0.00000...
64 14:12:52	SELECT * FROM EMPLOYEE WHERE hire_date > '2017-01-01' LIMIT 0, 1000	3 row(s) returned	0.00027 sec / 0.0000...

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% 67:32

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 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.supp...	8 row(s) returned	0.00045 sec / 0.0000...
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 / 0.0000...