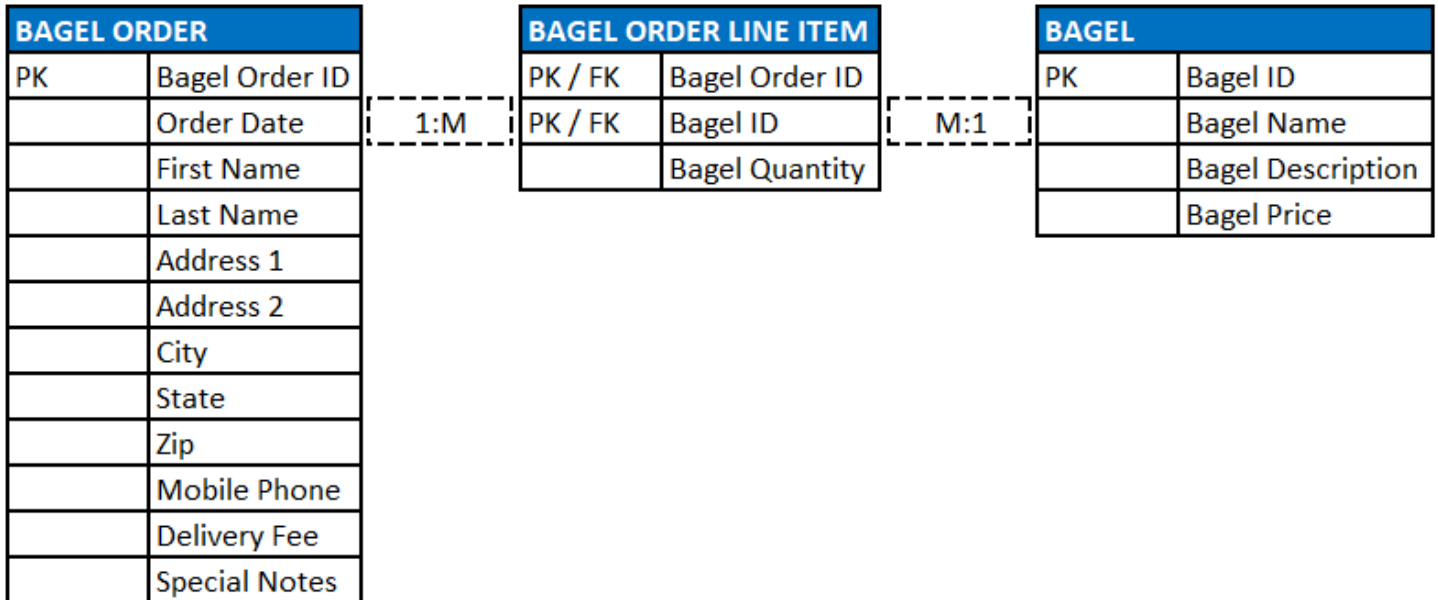


Part A: Nora's Bagel Bin Database Blueprints

A1a/A1b.

Second Normal Form (2NF)



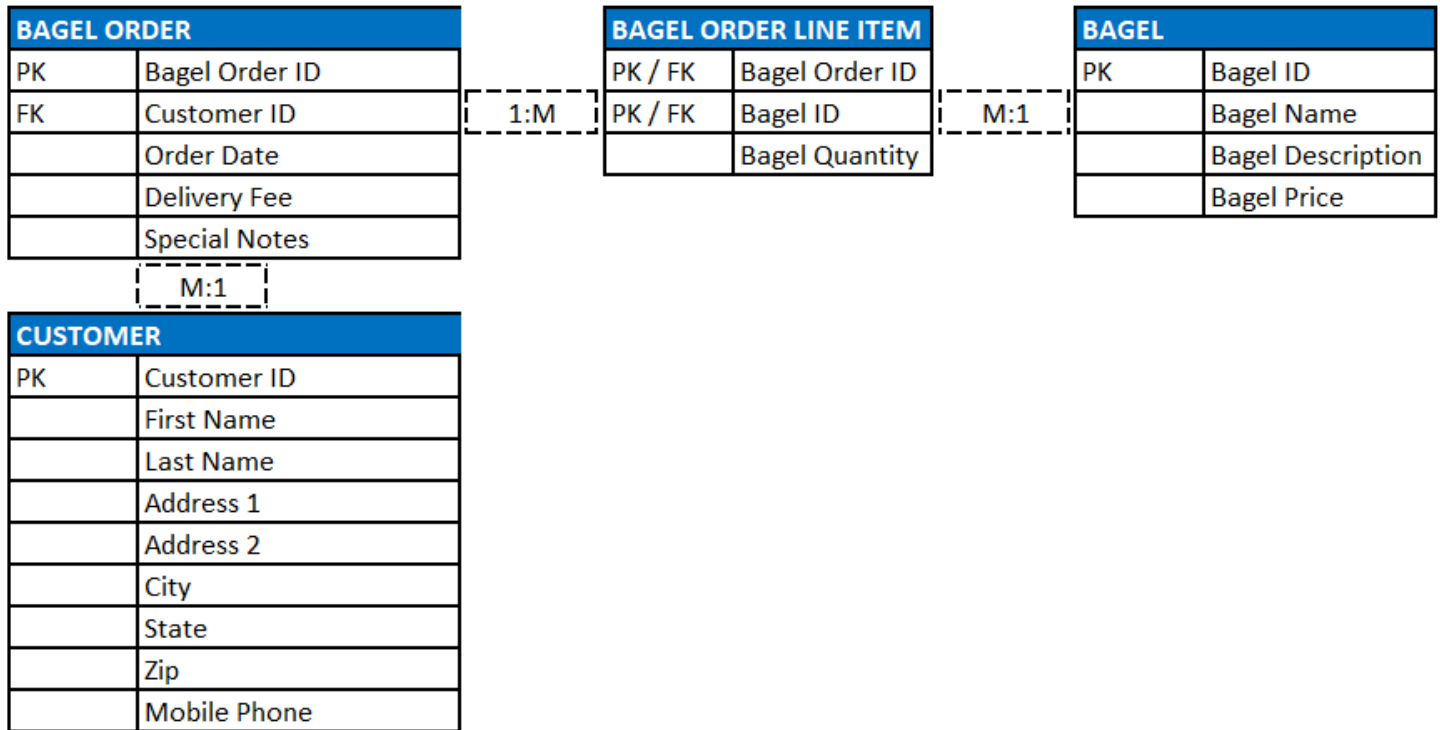
A1c.

The definition of second normal form says that all requirements of first normal form are met, and all non-key attributes must be functionally dependent on the entire primary key. The only attribute in the 1NF relation that depends on the entire primary key (*Bagel Order ID*, *Bagel ID*) is *Bagel Quantity*. Attributes *Bagel Name*, *Bagel Description*, and *Bagel Price* depend only on *Bagel ID*, but not *Bagel Order ID*, and can be moved to a separate relation. *Order Date*, *Delivery Fee*, *Special Notes*, and fields pertaining to customer information depend only on *Bagel Order ID*, but not *Bagel ID*, and can be moved to a separate relation. By removing the attributes not dependent on the entire primary key and placing them into separate relations in which they are dependent on the entire primary key, second normal form compliance is achieved.

Relationship cardinality can be determined by considering the actual ordering process. A Bagel Order can include many Bagel Order Line Items, while a Bagel Order Line Item belongs to one particular Bagel Order (1:M). A Bagel Order Line Item refers to one type of bagel, while a Bagel can belong to many Bagel Order Line Items(M:1).

A2a/A2b/A2c/A2d.

Third Normal Form (3NF)

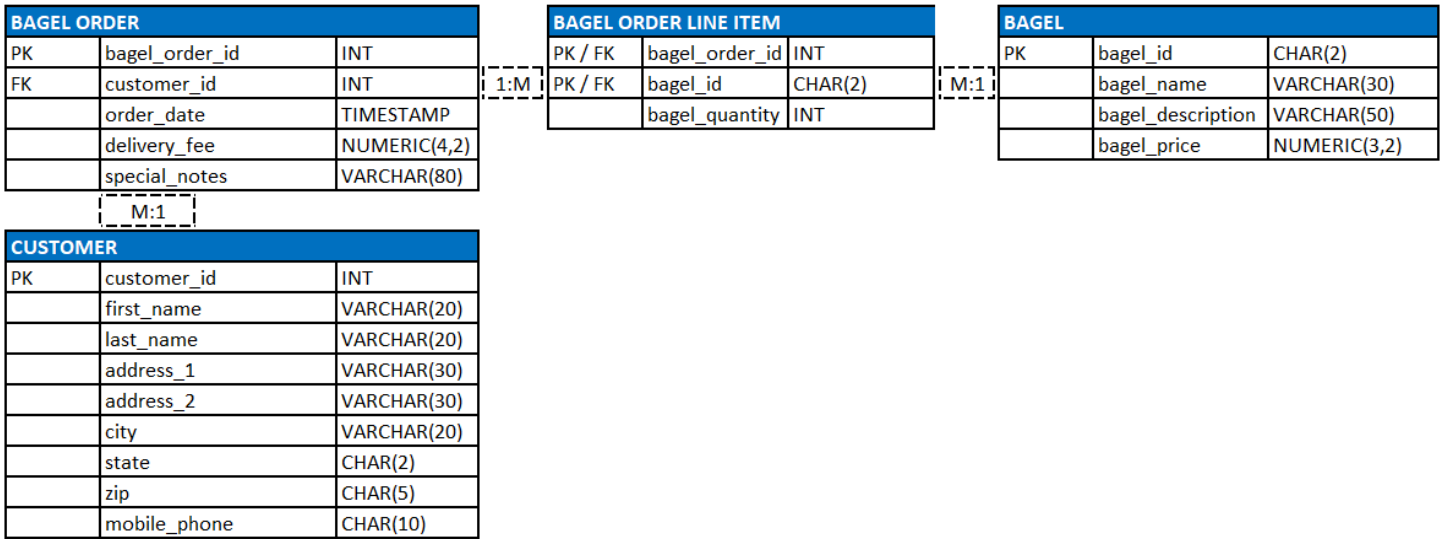
**A2e.**

The definition of third normal form says that all requirements of second normal form are met, and all non-key attributes depend only on the primary key. Attributes pertaining to customer information do depend on *Bagel Order ID*. However, those attributes also depend on the identity of the customer, or a *Customer ID* attribute. By removing the attributes that depend on *Customer ID* from the *Bagel Order* relation and placing them into a separate *Customer* relation, third normal form compliance is achieved.

The cardinality of the relationships between *Bagel Order* and *Customer* can be determined by considering the ordering process. A Bagel Order is placed by one individual Customer, while an individual Customer can place many Bagel Orders (M:1).

A3a/A3b.

Final Physical Database Model



Part B: Jaunty Coffee Co. Database

B1a (coffee_shop and employee table creation).

```
CREATE TABLE coffee_shop (  
  shop_id      INT,  
  shop_name    VARCHAR(50),  
  city         VARCHAR(50),  
  state        CHAR(2),  
  PRIMARY KEY (shop_id)  
);  
  
CREATE TABLE employee (  
  employee_id  INT,  
  first_name   VARCHAR(30),  
  last_name    VARCHAR(30),  
  hire_date    DATE,  
  job_title    VARCHAR(30),  
  shop_id      INT,  
  PRIMARY KEY (employee_id),  
  FOREIGN KEY (shop_id) REFERENCES coffee_shop (shop_id)  
);
```

B1b (coffee_shop and employee table creation).

The screenshot shows a database management interface with a left sidebar displaying a schema tree for 'jaunty'. The main area shows the SQL commands for creating the 'coffee_shop' and 'employee' tables. The 'Action Output' pane at the bottom shows the execution results of these commands.

Object Info

No object selected

Action Output

	Time	Action	Response
✓ 1	22:32:59	CREATE TABLE coffee_shop (shop_id INT, shop_name VARCHAR(50), city VARCHAR(50), state C...	0 row(s) affected
✓ 2	22:32:59	CREATE TABLE employee (employee_id INT, first_name VARCHAR(30), last_name VARCHAR(30), h...	0 row(s) affected

B1a (coffee and supplier table creation).

```
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 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)  
);
```

B1b (coffee and supplier table creation).

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' panel shows a tree view with 'jaunty' expanded, containing 'Tables' (coffee, coffee_shop, employee, supplier) and 'Views'. The main area displays SQL code for creating two tables: 'supplier' and 'coffee'. The 'supplier' table has columns: supplier_id (INTEGER, PRIMARY KEY), company_name (VARCHAR(50)), country (VARCHAR(30)), sales_contact_name (VARCHAR(60)), and email (VARCHAR(50) NOT NULL). The 'coffee' table has columns: coffee_id (INTEGER, PRIMARY KEY), shop_id (INTEGER, FOREIGN KEY to coffee_shop), supplier_id (INTEGER, FOREIGN KEY to supplier), coffee_name (VARCHAR(30)), and price_per_pound (NUMERIC(5,2)). The bottom panel shows the 'Action Output' table with two rows indicating successful table creation.

Object Info	Session	Action Output
No object selected		
Time	Action	Response
22:34:15	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))	0 row(s) affected
22:34:15	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))	0 row(s) affected

B2a (populate coffee_shop table).

```
INSERT INTO coffee_shop (shop_id, shop_name, city, state)
```

```
VALUES
```

```
(1131, 'Jaunty Coffee Las Vegas', 'Las Vegas', 'NV'),  
(1145, 'Jaunty Coffee Schaumburg', 'Schaumburg', 'IL'),  
(1146, 'Jaunty Coffee Glenview', 'Glenview', 'IL'),  
(1148, 'Jaunty Coffee Water Tower Place', 'Chicago', 'IL'),  
(1157, 'Jaunty Coffee Financial District', 'Chicago', 'IL'),  
(1201, 'Jaunty Coffee OHare', 'Chicago', 'IL');
```

B2b (populate coffee_shop table).

The screenshot shows a database management interface with a left sidebar containing a tree view of schemas. The 'jaunty' schema is expanded, showing tables like 'coffee', 'coffee_shop', 'employee', and 'supplier'. The main area displays the following SQL code:

```
1 INSERT INTO coffee_shop (shop_id, shop_name, city, state)  
2 VALUES  
3 (1131, 'Jaunty Coffee Las Vegas', 'Las Vegas', 'NV'),  
4 (1145, 'Jaunty Coffee Schaumburg', 'Schaumburg', 'IL'),  
5 (1146, 'Jaunty Coffee Glenview', 'Glenview', 'IL'),  
6 (1148, 'Jaunty Coffee Water Tower Place', 'Chicago', 'IL'),  
7 (1157, 'Jaunty Coffee Financial District', 'Chicago', 'IL'),  
8 (1201, 'Jaunty Coffee OHare', 'Chicago', 'IL');  
9
```

Below the SQL editor, there is a table with columns 'Object Info', 'Session', 'Action Output', and 'Response'. The 'Action Output' column shows the execution of the SQL statement, and the 'Response' column shows the results: 6 row(s) affected, Records: 6, Duplicates: 0, Warnings: 0.

Object Info	Session	Action Output	Response
No object selected		1 22:35:28 INSERT INTO coffee_shop (shop_id, shop_name, city, state) VALUES (1131, 'Jaunty Coffee Las Vega...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0

B2a (populate employee table).

```
INSERT INTO employee (employee_id, first_name, last_name, hire_date, job_title, shop_id)
VALUES
(130127, 'Daniel', 'Risco', '2020-11-01', 'Barista', 1145),
(143007, 'Dianne', 'Baylon', '2022-04-03', 'Shift Supervisor', 1148),
(147644, 'Evan', 'Berry', '2022-07-19', 'Barista', 1201),
(109311, 'Rebecca', 'Moeller', '2017-06-17', 'Dishwasher', 1148),
(148573, 'Coral', 'Rodriguez', '2022-09-23', 'Shift Supervisor', 1146),
(149589, 'Christina', 'Briseno', '2022-10-08', 'Barista', 1131);
```

B2b (populate employee table).

The screenshot shows a database management interface with a left sidebar containing a tree view of schemas. The 'jaunty' schema is expanded, showing tables like 'coffee', 'coffee_shop', 'employee', and 'supplier'. The main area displays the execution of an SQL statement. The statement is: `INSERT INTO employee (employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES (130127, 'Daniel', 'Risco', '2020-11-01', 'Barista', 1145), (143007, 'Dianne', 'Baylon', '2022-04-03', 'Shift Supervisor', 1148), (147644, 'Evan', 'Berry', '2022-07-19', 'Barista', 1201), (109311, 'Rebecca', 'Moeller', '2017-06-17', 'Dishwasher', 1148), (148573, 'Coral', 'Rodriguez', '2022-09-23', 'Shift Supervisor', 1146), (149589, 'Christina', 'Briseno', '2022-10-08', 'Barista', 1131);`. The bottom panel shows the 'Action Output' with a table containing one row: a green checkmark, the time '22:35:58', the truncated SQL statement, and the response '6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0'.

Object Info	Session	Action Output	Response						
No object selected		<table><thead><tr><th></th><th>Time</th><th>Action</th></tr></thead><tbody><tr><td>✓ 1</td><td>22:35:58</td><td>INSERT INTO employee (employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES...</td></tr></tbody></table>		Time	Action	✓ 1	22:35:58	INSERT INTO employee (employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0
	Time	Action							
✓ 1	22:35:58	INSERT INTO employee (employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES...							

B2a (populate supplier table).

```
INSERT INTO supplier (supplier_id, company_name, country, sales_contact_name, email)
VALUES
(6061134, 'Specialty Java', 'United States', 'Hayley Wright',
'hwright@specjav.com'),
(6061138, 'Volcanica Coffee', 'Colombia', 'Sarai Romano',
'sarai@volcanica.com'),
(6061141, 'Switch Coffee', 'Japan', 'Samantha
Anderson', 'anderson.s@switchcoffee.com'),
(6061167, 'Square Mile', 'England', 'Peter Davies',
'pete@squaremile.com'),
(6061183, 'Coffee Supreme', 'New Zealand', 'Jemaine Clement',
'jclement@supremenz.com')
```

B2b (populate supplier table).

The screenshot shows a database management interface with a left sidebar containing a tree view of schemas. The 'jaunty' schema is selected, showing a list of tables including 'coffee', 'coffee_shop', 'employee', 'supplier', 'Views', 'Stored Procedures', 'Functions', and 'sys'. The main area displays the execution of an SQL statement. The statement is: `INSERT INTO supplier (supplier_id, company_name, country, sales_contact_name, email) VALUES (6061134, 'Specialty Java', 'United States', 'Hayley Wright', 'hwright@specjav.com'), (6061138, 'Volcanica Coffee', 'Colombia', 'Sarai Romano', 'sarai@volcanica.com'), (6061141, 'Switch Coffee', 'Japan', 'Samantha Anderson', 'anderson.s@switchcoffee.com'), (6061167, 'Square Mile', 'England', 'Peter Davies', 'pete@squaremile.com'), (6061183, 'Coffee Supreme', 'New Zealand', 'Jemaine Clement', 'jclement@supremenz.com')`. The bottom status bar indicates that the statement was executed successfully at 22:36:13, affecting 5 rows, with 0 duplicates and 0 warnings.

Object Info	Session	Action Output						
No object selected								
		<table border="1"><thead><tr><th>Time</th><th>Action</th><th>Response</th></tr></thead><tbody><tr><td>22:36:13</td><td>INSERT INTO supplier (supplier_id, company_name, country, sales_contact_name, email) VALUES (...)</td><td>5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0</td></tr></tbody></table>	Time	Action	Response	22:36:13	INSERT INTO supplier (supplier_id, company_name, country, sales_contact_name, email) VALUES (...)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
Time	Action	Response						
22:36:13	INSERT INTO supplier (supplier_id, company_name, country, sales_contact_name, email) VALUES (...)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0						

B2a (populate coffee table).

```
INSERT INTO coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
VALUES
(8001, 1131, 6061134, 'Signature Blend', 13.23),
(8003, 1145, 6061138, 'Fazenda Capadocia', 12.14),
(8004, 1146, 6061141, 'Gathaithi', 12.88),
(8010, 1148, 6061167, 'Terra Nova', 14.00),
(8016, 1157, 6061183, 'Rwanda Tumba', 11.90),
(8033, 1201, 6061138, 'Los Andes', 14.20)
```

B2b (populate coffee table).

The screenshot shows a database management interface with a left sidebar for 'SCHEMAS' and a main editor area. The sidebar lists a schema named 'jaunty' containing several tables: 'coffee', 'coffee_shop', 'employee', 'supplier', 'Views', 'Stored Procedures', and 'Functions'. The main editor displays the following SQL code:

```
1 INSERT INTO coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound)
2 VALUES
3 (8001, 1131, 6061134, 'Signature Blend', 13.23),
4 (8003, 1145, 6061138, 'Fazenda Capadocia', 12.14),
5 (8004, 1146, 6061141, 'Gathaithi', 12.88),
6 (8010, 1148, 6061167, 'Terra Nova', 14.00),
7 (8016, 1157, 6061183, 'Rwanda Tumba', 11.90),
8 (8033, 1201, 6061138, 'Los Andes', 14.20)
9
```

Below the editor, the 'Action Output' tab is active, showing the execution results:

	Time	Action	Response
✓ 1	22:36:26	INSERT INTO coffee (coffee_id, shop_id, supplier_id, coffee_name, price_per_pound) VALUES (800...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0

B3a.

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

B3b.

The screenshot displays the SQL Developer interface. The left sidebar shows the 'SCHEMAS' tree with 'jaunty' expanded, containing 'Tables' (coffee, coffee_shop, employee, supplier) and 'Views' (employee_view). The main editor shows the SQL script for creating the view. The 'Action Output' pane shows the successful execution of the 'CREATE VIEW' statement. Below, the 'Result Grid' displays the data returned by the 'SELECT * FROM employee_view' query.

SCHEMAS

Filter objects

- jaunty
 - Tables
 - coffee
 - coffee_shop
 - employee
 - supplier
 - Views
 - employee_view
 - Stored Procedures
 - Functions
- sys

Object Info | **Session**

No object selected

SCHEMAS

Filter objects

- jaunty
 - Tables
 - coffee
 - coffee_shop
 - employee
 - supplier
 - Views
 - employee_view
 - Stored Procedures
 - Functions
- sys

Object Info | **Session**

No object selected

Action Output

	Time	Action	Response
1	22:51:14	CREATE VIEW employee_view AS SELECT employee_id, CONCAT(first_name, " ", last_name)...	0 row(s) affected

Result Grid | Filter Rows: | Search | Export:

employee_id	employee_full_name	hire_date	job_title	shop_id
109311	Rebecca Moeller	2017-06-17	Dishwasher	1148
130127	Daniel Risco	2020-11-01	Barista	1145
143007	Dianne Baylon	2022-04-03	Shift Supervisor	1148
147644	Evan Berry	2022-07-19	Barista	1201
148573	Coral Rodriguez	2022-09-23	Shift Supervisor	1146
149589	Christina Briseno	2022-10-08	Barista	1131

Object Info | **Session**

No object selected

Action Output

	Time	Action	Response
1	22:51:14	CREATE VIEW employee_view AS SELECT employee_id, CONCAT(first_name, " ", last_name)...	0 row(s) affected
2	22:52:02	SELECT * FROM employee_view LIMIT 0, 1000	6 row(s) returned

B4a.

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

B4b.

The screenshot shows a database client interface with a sidebar on the left displaying a schema tree for 'jaunty'. The main editor area contains the SQL statement: `CREATE INDEX coffee_name_index ON coffee (coffee_name);`. Below the editor, the 'Action Output' tab is active, showing a table with columns 'Time', 'Action', and 'Response'. The output shows a successful execution of the CREATE INDEX statement at 22:55:06, with 0 rows affected, 0 duplicates, and 0 warnings.

	Time	Action	Response
1	22:55:06	CREATE INDEX coffee_name_index ON coffee (coffee_name)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

B5a.

```
SELECT shop_id, shop_name, city
FROM coffee_shop
WHERE state='IL';
```

B5b.

The screenshot shows the same database client interface. The main editor area contains the SQL statement: `SELECT shop_id, shop_name, city FROM coffee_shop WHERE state='IL';`. Below the editor, the 'Result Grid' tab is active, displaying a table with columns 'shop_id', 'shop_name', and 'city'. The table contains 5 rows of data, including shop information for Schaumburg, Glenview, and Chicago. The 'Action Output' tab at the bottom shows the execution of the SELECT statement at 23:00:14, returning 5 rows.

shop_id	shop_name	city
1145	Jaunty Coffee Schaumburg	Schaumburg
1146	Jaunty Coffee Glenview	Glenview
1148	Jaunty Coffee Water Tower Place	Chicago
1157	Jaunty Coffee Financial District	Chicago
1201	Jaunty Coffee OHare	Chicago
NULL	NULL	NULL

	Time	Action	Response
1	23:00:14	SELECT shop_id, shop_name, city FROM coffee_shop WHERE state='IL' LIMIT 0, 1000	5 row(s) returned

B6a.

```
SELECT coffee_shop.shop_id, coffee.coffee_name, supplier.sales_contact_name,  
supplier.email  
FROM coffee  
JOIN coffee_shop ON coffee.shop_id=coffee_shop.shop_id  
JOIN supplier ON coffee.supplier_id=supplier.supplier_id
```

B6b.

The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' pane lists the database 'jaunty' with its tables: 'coffee', 'coffee_shop', 'employee', and 'supplier'. The main area displays a SQL query:

```
1 SELECT coffee_shop.shop_id, coffee.coffee_name, supplier.sales_contact_name, supplier.email  
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  
5
```

Below the query, a 'Result Grid' shows the results of the query. The grid has columns: 'shop_id', 'coffee_name', 'sales_contact_name', and 'email'. There are 6 rows of data.

shop_id	coffee_name	sales_contact_name	email
1131	Signature Blend	Hayley Wright	hwright@specjav.com
1145	Fazenda Capadocia	Sarai Romano	sarai@volcanica.com
1201	Los Andes	Sarai Romano	sarai@volcanica.com
1146	Gathaithi	Samantha Anderson	anderson.s@switchcoffee.com
1148	Terra Nova	Peter Davies	pete@squaremile.com
1157	Rwanda Tumba	Jemaine Clement	jclement@supremenz.com

At the bottom, the 'Action Output' pane shows the execution of the query, indicating that 6 rows were returned.

	Time	Action	Response
✓ 1	23:13:29	SELECT coffee_shop.shop_id, coffee.coffee_name, supplier.sales_contact_name, supplier.email FR...	6 row(s) returned