PART A: Nora's Bagel Bin Database Blueprints

First Normal Form (1NF)

BAGEL ORDER	
PK	Bagel Order ID
PK	Bagel ID
	Order Date
	First Name
	Last Name
	Address 1
	Address 2
	City
	State
	Zip
	Mobile Phone
	Delivery Fee
	Bagel Name
	Bagel Description
	Bagel Price
	Bagel Quantity
	Special Notes

Second Normal Form (2NF)

BAGEL ORDER	
PK	Bagel Order ID
	Order Date
	First Name
	Last Name
	Address 1
	Address 2
	City
	State
	Zip
	Mobile Phone
	Delivery Fee
	Special Notes

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

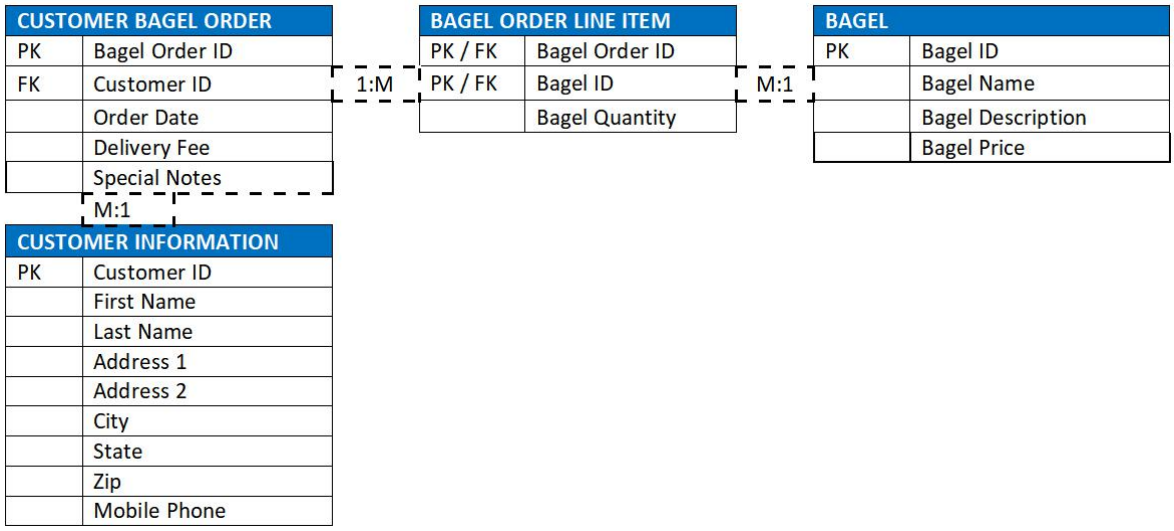
With (Bagel Order ID, Bagel ID) being the primary key for the Bagel Order Line Item the only attribute that is dependent on both aspects of the primary key is Bagel Quantity. In order to know the particular quantity of a particular bagel that was ordered in any particular BAGEL ORDER you need to know what Bagel Order ID and which Bagel in particular was ordered, making it functionally dependent on the entire primary key of Bagel Order Line Item.

Bagel Name, Bagel Description, and Bagel Price depend entirely on what Bagel ID is associated with it, and have thus been placed in its own BAGEL table, with Bagel ID as a primary key that is linked to BAGEL ORDER LINE ITEM by the Bagel ID foreign key. The rest of the fields rely solely on the particular BAGEL ORDER that they are attached to and have been placed into the BAGEL ORDER table.

For cardinality, BAGEL ORDER maintains a 1:M relationship with BAGEL ORDER LINE ITEM because each BAGEL ORDER can contain multiple BAGEL ORDER LINE ITEM's, but each specific BAGEL ORDER LINE ITEM can only be associated with one BAGEL ORDER since the primary key of BAGEL ORDER LINE ITEM contains within its partial

primary key the Bagel Order ID. BAGEL ORDER LINE ITEM has a M:1 relationship with BAGEL because each line within BAGEL ORDERLINE ITEM is associated with only one Bagel ID while many different bagels can be included in the BAGEL ORDER LINE ITEM table.

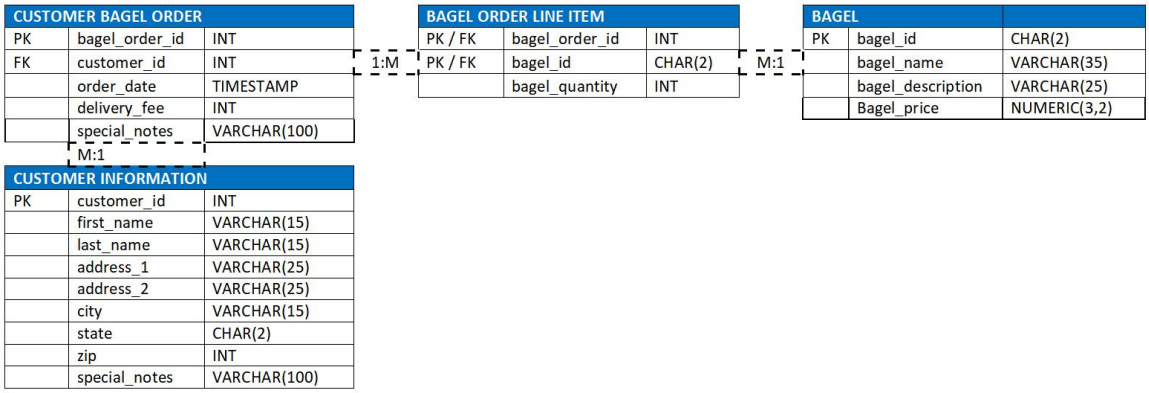
Third Normal Form (3NF)



First Name, Last Name, Address 1, Address_2, City, State, Zip, and Mobile Phone do not depend on the primary key Bagel Order ID, but rather on a particular Customer and have been moved into their own table labeled CUSTOMER INFORMATION. The BAGEL ORDER table has been renamed CUSTOMER BAGEL ORDER to reflect that it is now an order for a particular customer. A new attribute titled Customer ID has been created and placed in CUSTOMER BAGEL ORDER as a foreign key and in CUSTOMER INFORMATION as a primary key to link the two tables.

A customer may have many individual bagel orders throughout the life of the business, and each individual customer bagel order pertains to only one customer in particular. This makes the relationship between CUSTOMER BAGEL ORDER and CUSTOMER INFORMATION a M:1 relationship. The other relationships carried over from 2NF have remained the same.

Final Physical Database Model

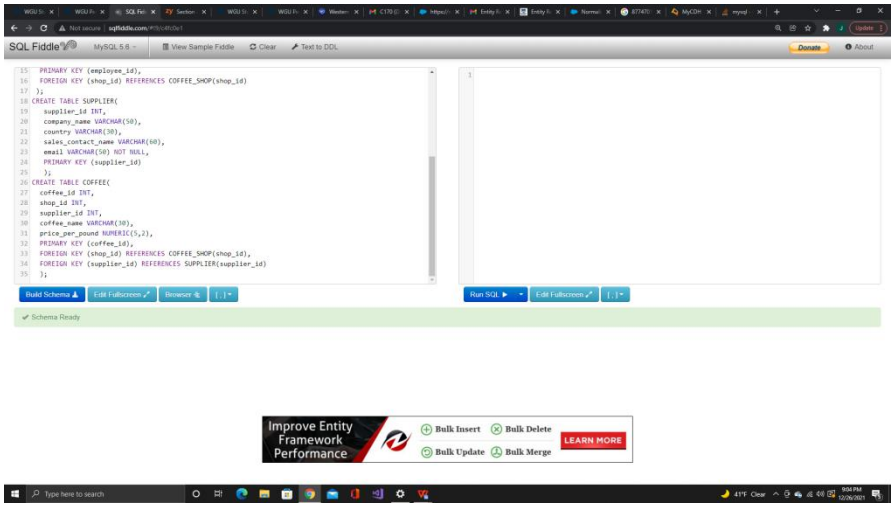


PART B: Jauntey Coffee Database

1. Develop SQL code to create *each* table as specified in the attached "Jaunty Coffee Co. ERD".

SQL CODE:

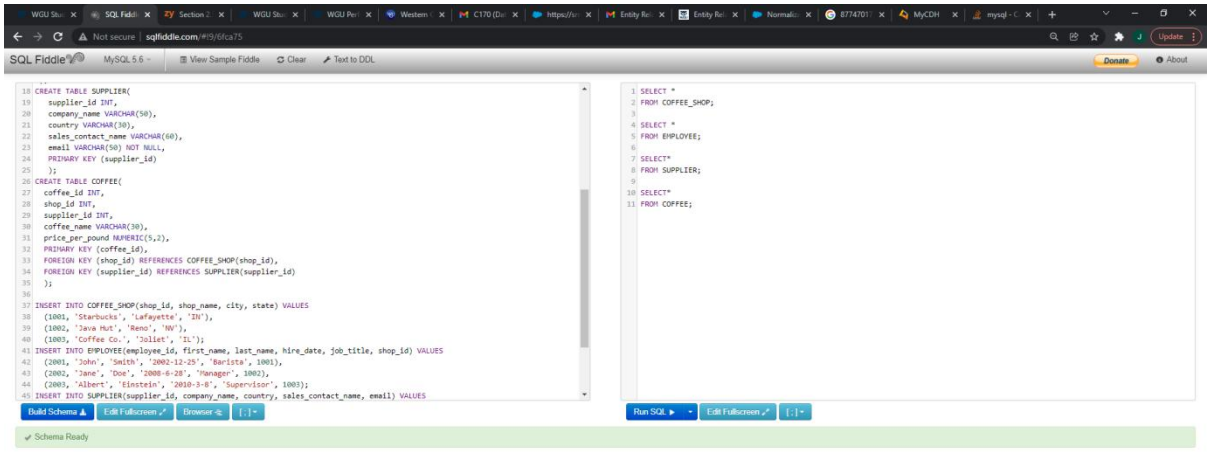
```
CREATE TABLE COFFEE_SHOP(
  shop_id INT,
  shop_name VARCHAR(50),
  city VARCHAR(50),
  state CHAR(2),
  PRIMARY KEY (shop_id)
 );
CREATE TABLE EMPLOYEE(
  employee_id INT,
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  hire_date DATE,
  job_title VARCHAR(30),
  shop_id INT,
  PRIMARY KEY (employee_id),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

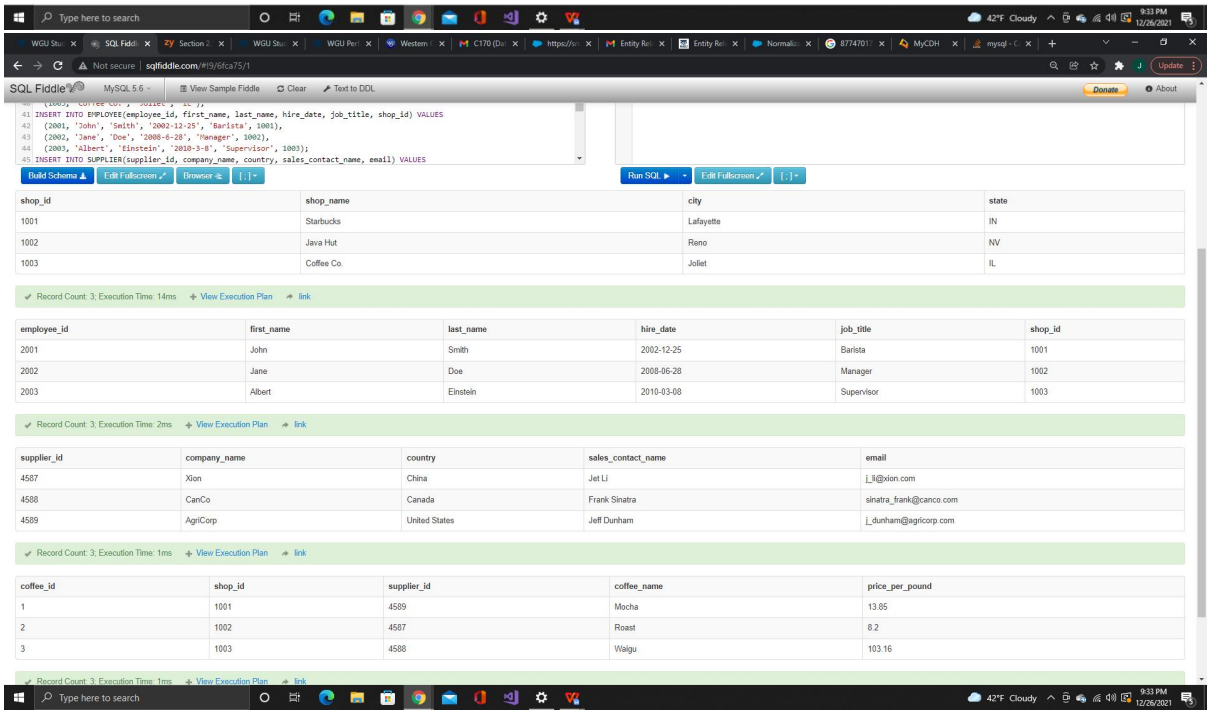


2. Develop SQL code to populate *each* table in the database design document.

SQL CODE:

```
INSERT INTO COFFEE_SHOP(shop_id, shop_name, city, state) VALUES
  (1001, 'Starbucks', 'Lafayette', 'IN'),
  (1002, 'Java Hut', 'Reno', 'NV'),
  (1003, 'Coffee Co.', 'Joliet', 'IL');
INSERT INTO EMPLOYEE(employee_id, first_name, last_name, hire_date, job_title, shop_id) VALUES
  (2001, 'John', 'Smith', '2002-12-25', 'Barista', 1001),
  (2002, 'Jane', 'Doe', '2008-6-28', 'Manager', 1002),
  (2003, 'Albert', 'Einstein', '2010-3-8', 'Supervisor', 1003);
INSERT INTO SUPPLIER(supplier_id, company_name, country, sales_contact_name, email) VALUES
  (4587, 'Xion', 'China', 'Jet Li', 'j_li@xion.com'),
  (4588, 'CanCo', 'Canada', 'Frank Sinatra', 'sinatra_frank@canco.com'),
  (4589, 'AgriCorp', 'United States', 'Jeff Dunham', 'j_dunham@agricorp.com');
INSERT INTO COFFEE(coffee_id, shop_id, supplier_id, coffee_name, price_per_pound) VALUES
  (01, 1001, 4589, 'Mocha', 13.85),
  (02, 1002, 4587, 'Roast', 8.20),
  (03, 1003, 4588, 'Waigu', 103.16);
```

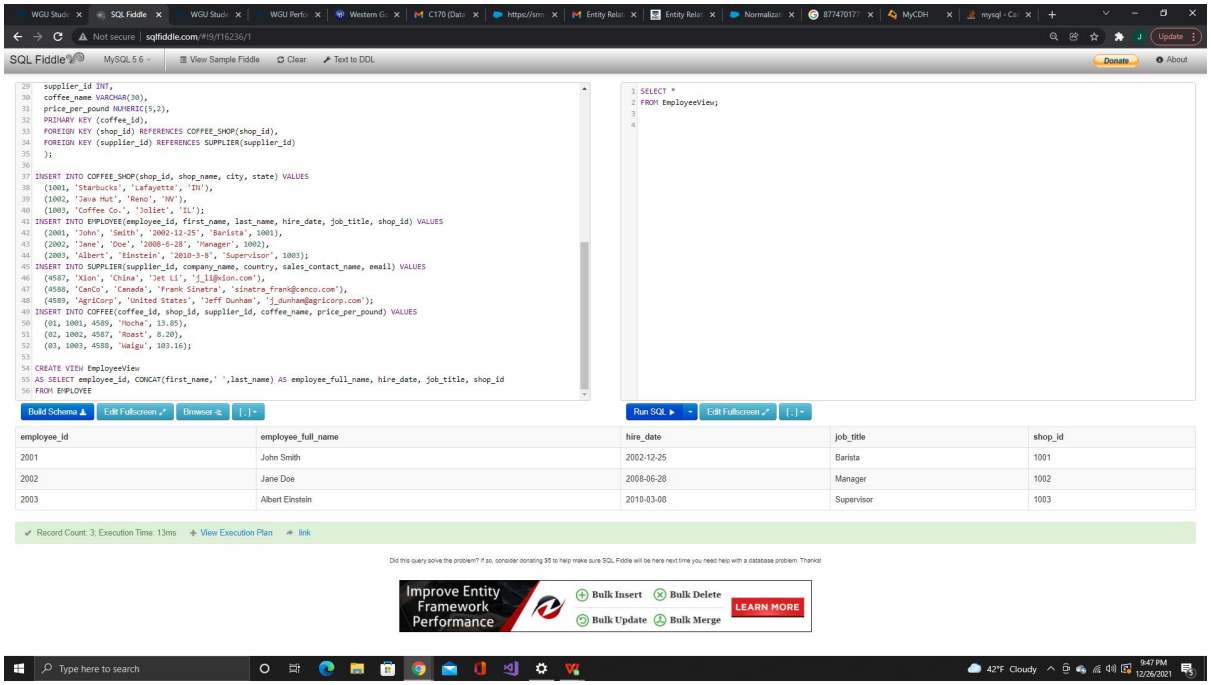




3. Develop SQL code to create a view.

SQL CODE:

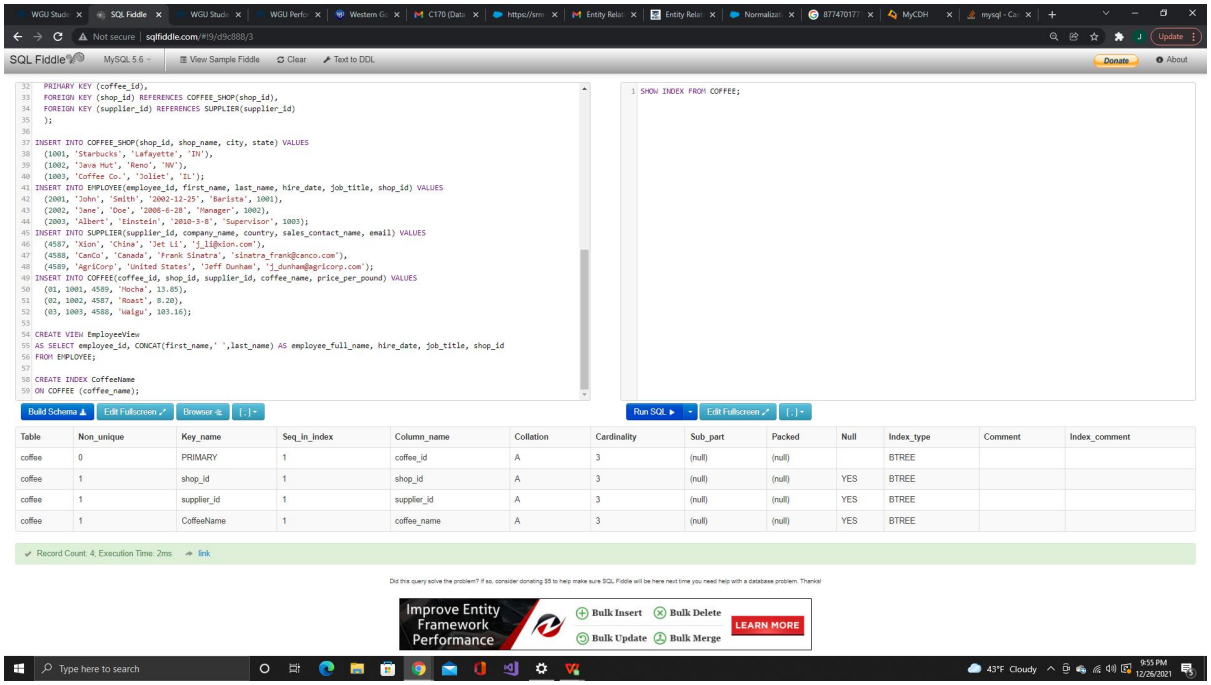
```
CREATE VIEW EmployeeView
AS SELECT employee_id, CONCAT(first_name,' ',last_name) AS employee_full_name, hire_date,
job_title, shop_id
FROM EMPLOYEE;
```



4. Develop SQL code to create an index on the coffee_name field.

SQL CODE:

```
CREATE INDEX CoffeeName
ON COFFEE (coffee_name);
```



5. Develop SQL code to create an SFW (SELECT–FROM–WHERE) query for *any* of your tables or views.

SQL CODE:

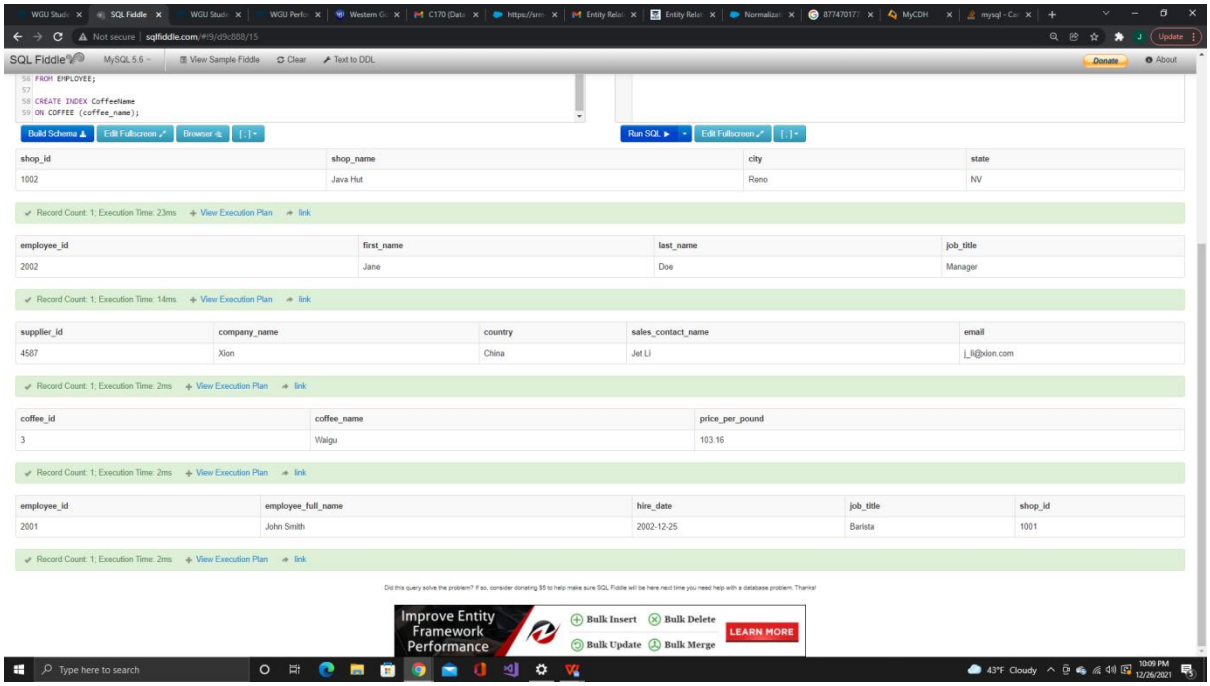
```
SELECT *
FROM COFFEE_SHOP
WHERE shop_id = 1002;

SELECT employee_id, first_name, last_name, job_title
FROM EMPLOYEE
WHERE job_title = 'Manager';

SELECT *
FROM SUPPLIER
WHERE country = 'China';

SELECT coffee_id, coffee_name, price_per_pound
FROM COFFEE
WHERE supplier_id = 4588;

SELECT *
FROM EmployeeView
WHERE employee_full_name = 'John Smith';
```



6. Develop SQL code to create a query.

SQL CODE:

```
SELECT *
FROM COFFEE_SHOP
JOIN COFFEE
ON COFFEE_SHOP.shop_id = COFFEE.shop_id
JOIN SUPPLIER
ON COFFEE.supplier_id = SUPPLIER.supplier_id;
```