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

Having been provided the 1NF of Nora’s bagel ordering process there exists a composite key indicating multiple entities in the table. Because of this the data is functionally dependent on multiple entities. It is necessary to perform normalization of the table above to eliminate data redundancy and improve functionality.

Nora’s Bagel Bin Database Blueprints *(continued)*

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

1:M

BAGEL	
PK	bagel_id
	bagel_name
	bagel_description
	bagel_price

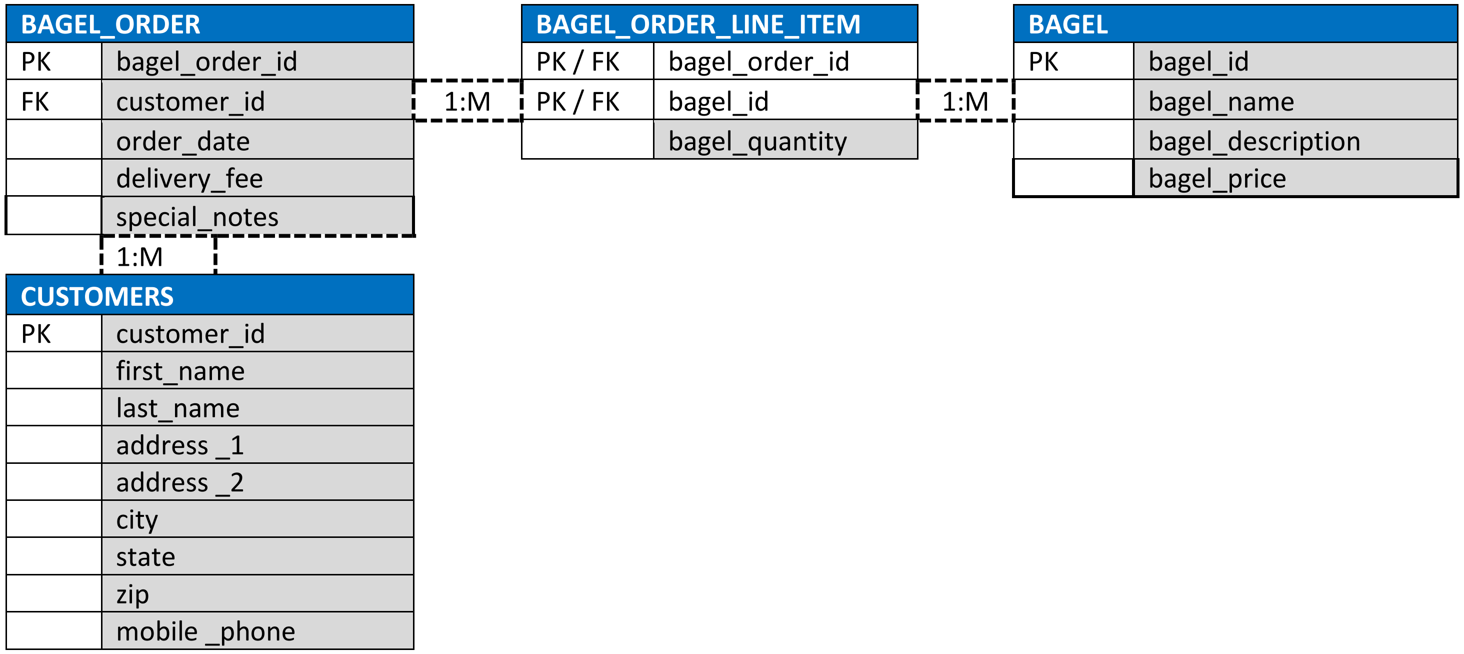
Achieving second normal form the initial entity has been separated into three distinct entities BAGEL_ORDER, BAGEL_ORDER_LINE_ITEM, and BAGEL. The BAGEL_ORDER entity containing the bagel_order_id primary key contains information that will be most likely be unique to each order submitted of the customer’s information with bagel_order_id, order_date, etc. The BAGEL entity containing the bagel_id primary key stores information relating to the products offered by Nora. As the information of the products offered will not change or have very little chance to change from order to order it was best to separate it into its own. The BAGEL_ORDER_LINE_ITEM table acts as a bridge between the BAGEL_ORDER and BAGEL entities being the associate table that links the inherent many-to-many relationship between the two. The bagel_quantity field exists within the associate table so that the quantity is dependent on each new order, while also maintaining the association to the BAGEL entity it is related to.

The relationships between each entity were determined by:

- Each BAGEL_ORDER will have many BAGEL_ORDER_LINE_ITEMs, but each is associated with one specific order relying on the bagel_order_id establishing the 1:M relationship between BAGEL_ORDER and BAGEL_ORDER_LINE_ITEM.
- Similarly, though each BAGEL_ORDER_LINE_ITEM may be associated with many different bagel products, the bagel_id and bagel_quantity ensure that each instance of a line item will be independent of the bagels products available.

Nora’s Bagel Bin Database Blueprints *(continued)*

Third Normal Form (3NF)



BAGEL_ORDER	
PK	bagel_order_id
FK	customer_id
	order_date
	delivery_fee
	special_notes

BAGEL_ORDER_LINE_ITEM	
PK / FK	bagel_order_id
PK / FK	bagel_id
	bagel_quantity

BAGEL	
PK	bagel_id
	bagel_name
	bagel_description
	bagel_price

CUSTOMERS	
PK	customer_id
	first_name
	last_name
	address _1
	address _2
	city
	state
	zip
	mobile _phone

Achieving third normal form the entities in the second normal form still exist with the addition of the BAGEL_ORDER entity being further broken down to separate customer information into its own CUSTOMERS entity. One customer can place many different orders, but each order is associated with only one customer. Creating the CUSTOMERS entity allows for it to be independent of every order, reducing data redundancy from duplicate customer information in BAGEL_ORDER for every new order of the same customer with the second normal form database previously.

Nora's Bagel Bin Database Blueprints *(continued)*

Final Physical Database Model

BAGEL_ORDER		
PK	bagel_order_id	INT
FK	customer_id	INT
	order_date	TIMESTAMP
	delivery_fee	NUMERIC(2,2)
	special_notes	VARCHAR(255)

1:M

BAGEL_ORDER_LINE_ITEM		
PK / FK	bagel_order_id	INT
PK / FK	bagel_id	CHAR(2)
	bagel_quantity	INT

1:M

BAGEL		
PK	bagel_id	INT
	bagel_name	VARCHAR(60)
	bagel_descriptio n	VARCHAR(255)
	bagel_price	NUMERIC(2,2)

1:M

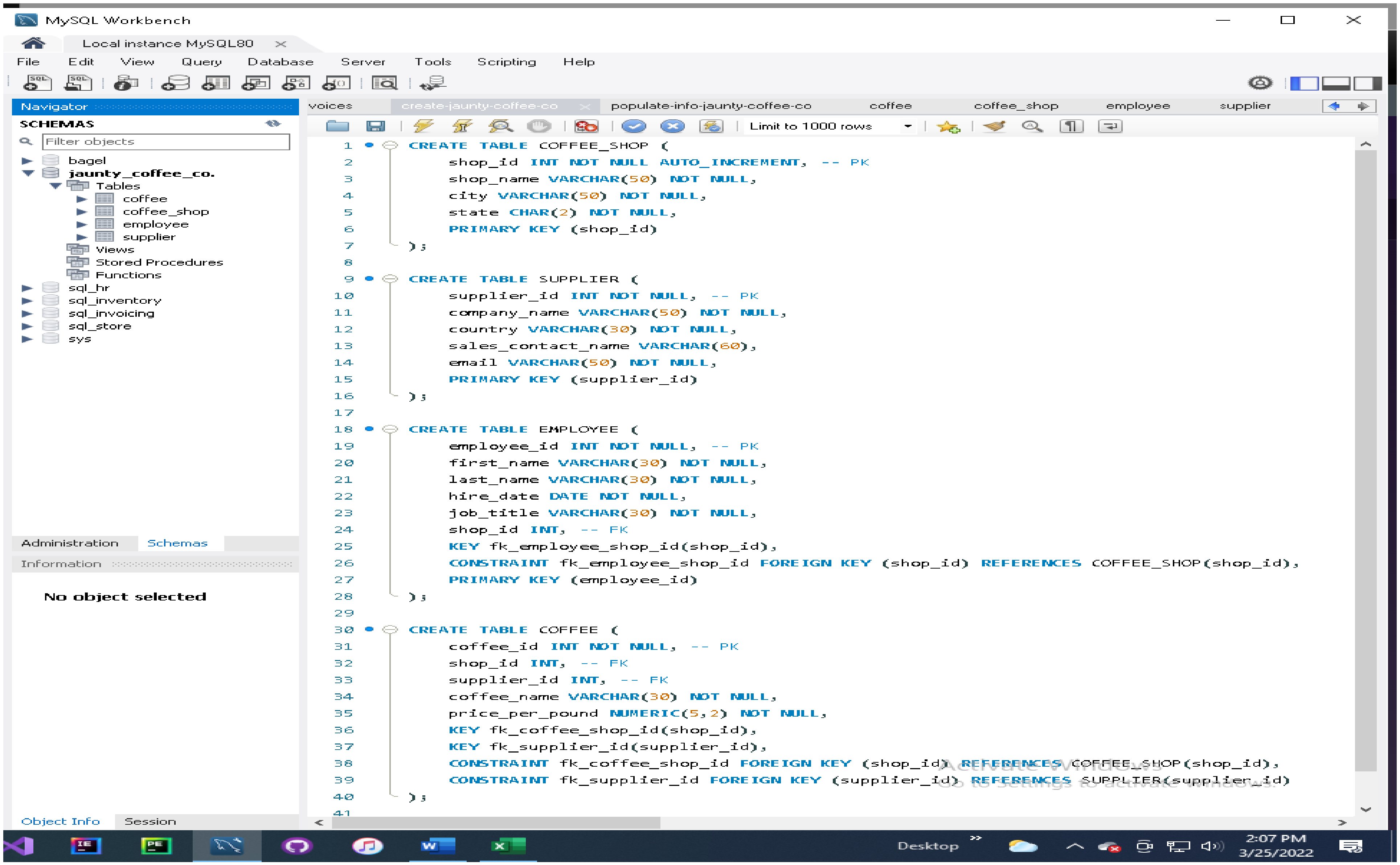
CUSTOMERS		
PK	customer_id	INT
	first_name	VARCHAR(60)
	last_name	VARCHAR(60)
	address_1	VARCHAR(60)
	address_2	VARCHAR(60)
	city	VARCHAR(60)
	state	CHAR(2)
	zip	CHAR(5)
	mobile_phone	VARCHAR(20)

The final database model reflects the changes made in the third normal form with the appropriate datatypes assigned.

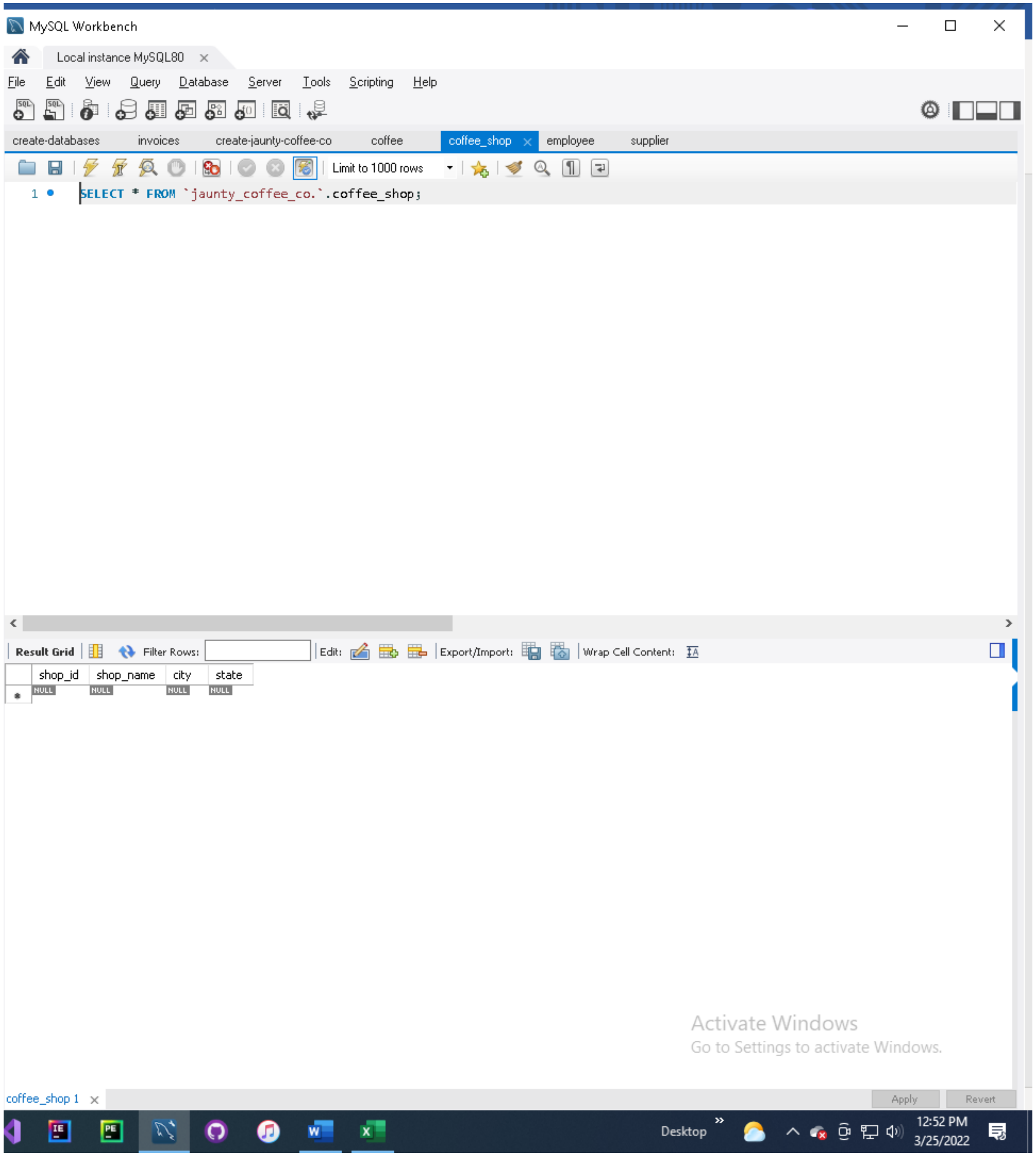
PART B

Jaunty Coffee Co. ERD

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

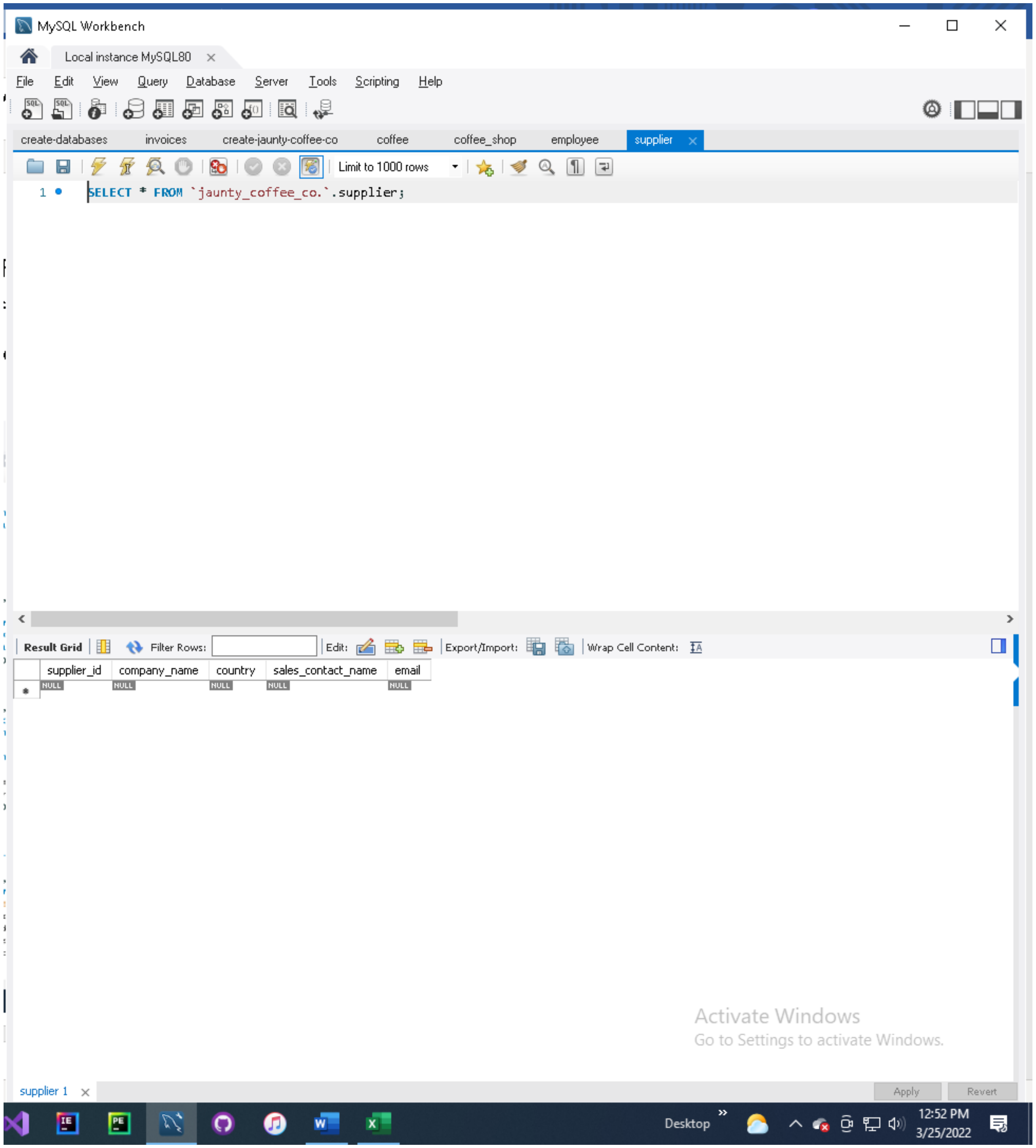


```sql
CREATE TABLE COFFEE_SHOP (
    shop_id INT NOT NULL AUTO_INCREMENT, -- PK
    shop_name VARCHAR(50) NOT NULL,
    city VARCHAR(50) NOT NULL,
    state CHAR(2) NOT NULL,
    PRIMARY KEY (shop_id)
);

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

CREATE TABLE EMPLOYEE (
    employee_id INT NOT NULL, -- PK
    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, -- FK
    KEY fk_employee_shop_id(shop_id),
    CONSTRAINT fk_employee_shop_id FOREIGN KEY (shop_id) REFERENCES COFFEE_SHOP(shop_id),
    PRIMARY KEY (employee_id)
);

CREATE TABLE COFFEE (
    coffee_id INT NOT NULL, -- PK
    shop_id INT, -- FK
    supplier_id INT, -- FK
    coffee_name VARCHAR(30) NOT NULL,
    price_per_pound NUMERIC(5,2) NOT NULL,
    KEY fk_coffee_shop_id(shop_id),
    KEY fk_supplier_id(supplier_id),
    CONSTRAINT fk_coffee_shop_id FOREIGN KEY (shop_id) REFERENCES COFFEE_SHOP(shop_id),
    CONSTRAINT fk_supplier_id FOREIGN KEY (supplier_id) REFERENCES SUPPLIER(supplier_id)
);
```



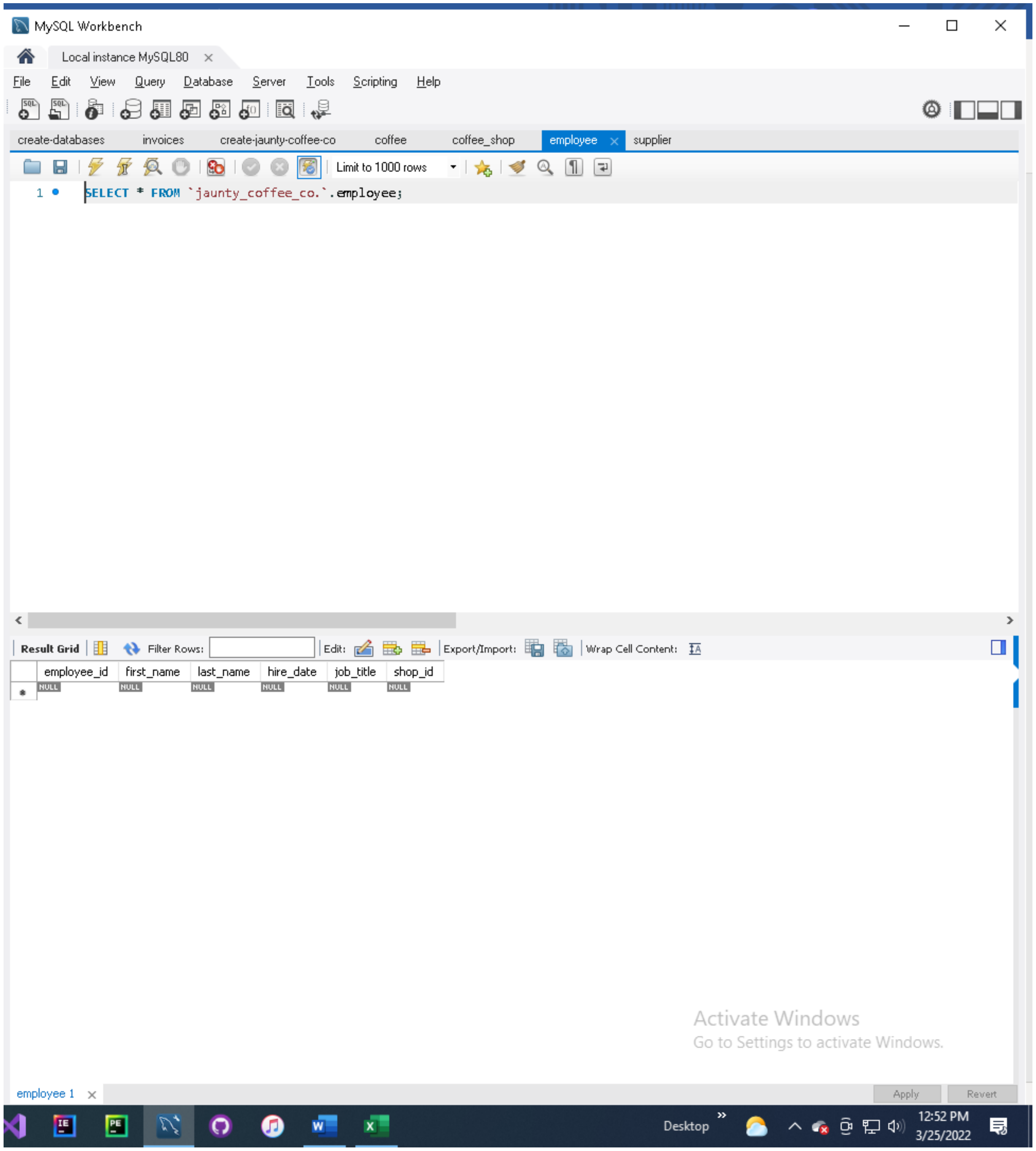
```
SELECT * FROM `jaunty_coffee_co.`.coffee_shop;
```

	shop_id	shop_name	city	state
*	NULL	NULL	NULL	NULL



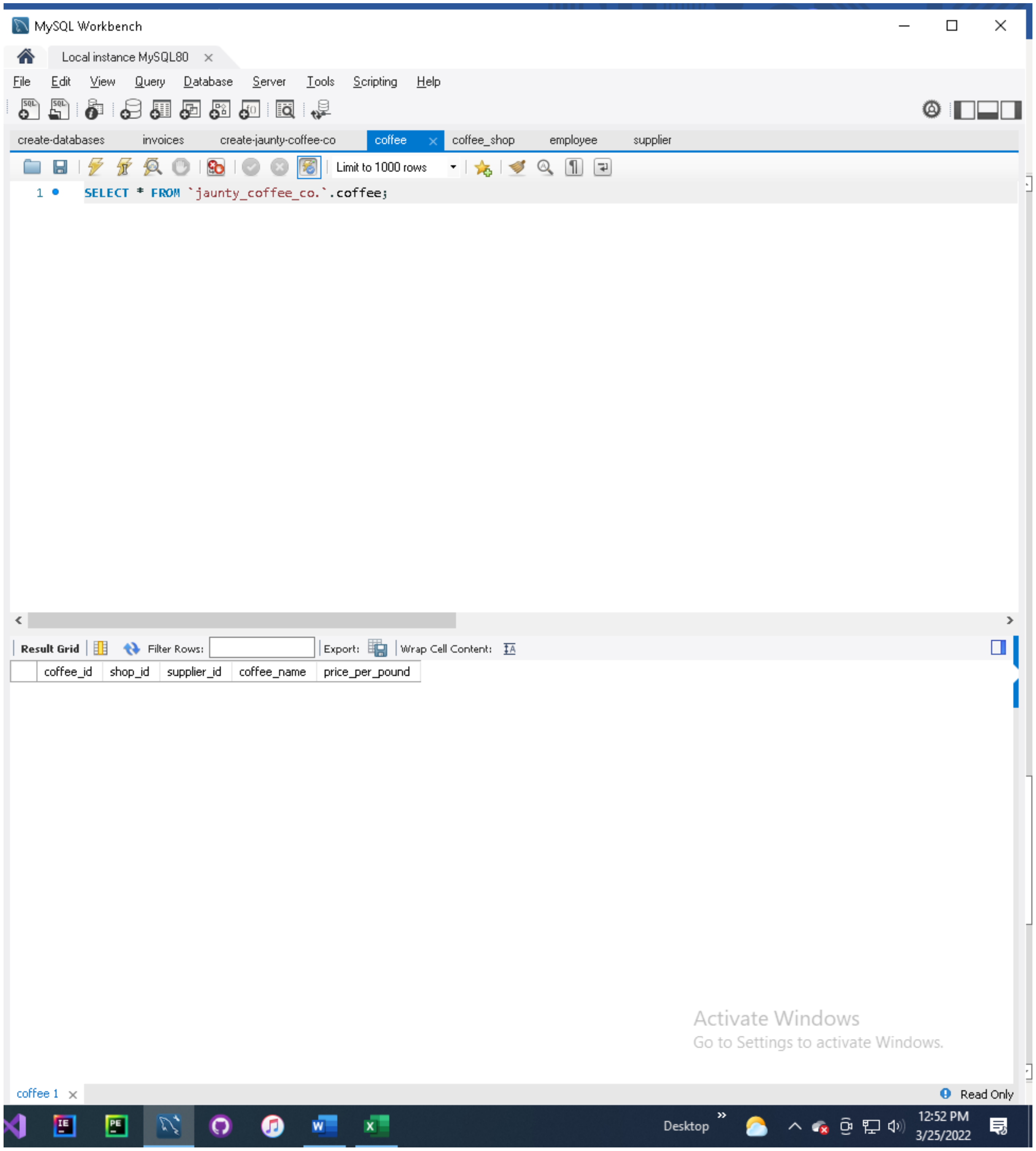
```
SELECT * FROM `jaunty_coffee_co.`.supplier;
```

	supplier_id	company_name	country	sales_contact_name	email
*	NULL	NULL	NULL	NULL	NULL


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
create-databases invoices create-jaunty-coffee-co coffee coffee_shop employee supplier
Limit to 1000 rows
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
employee 1
Apply Revert
Activate Windows
Go to Settings to activate Windows.
Desktop
12:52 PM
3/25/2022

```sql
SELECT * FROM `jaunty_coffee_co.`.employee;
```

	employee_id	first_name	last_name	hire_date	job_title	shop_id
*	NULL	NULL	NULL	NULL	NULL	NULL

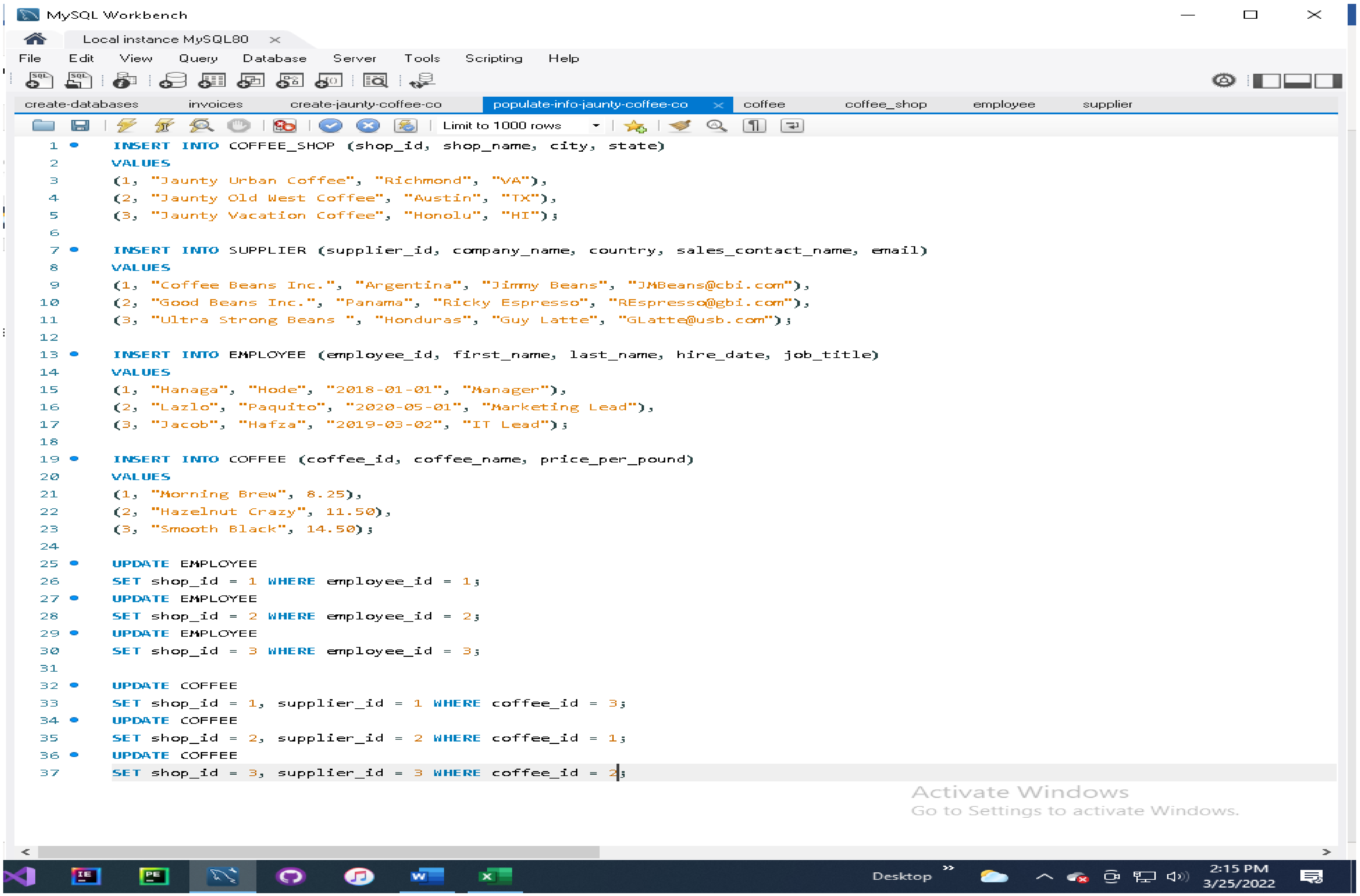


```
SELECT * FROM `jaunty_coffee_co.`.coffee;
```

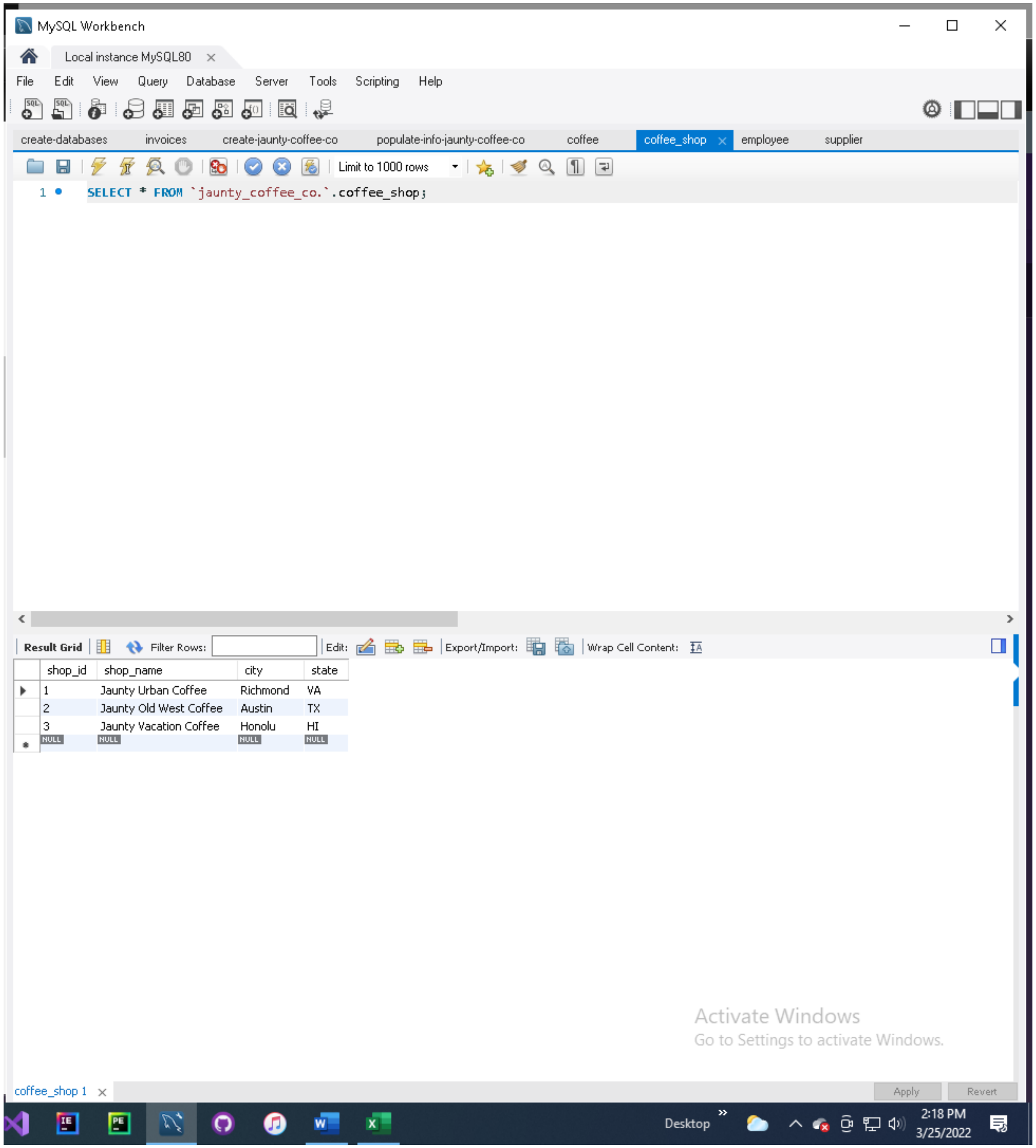
coffee_id	shop_id	supplier_id	coffee_name	price_per_pound

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
create-databases invoices create-jaunty-coffee-co coffee coffee_shop employee supplier
Limit to 1000 rows
Result Grid Filter Rows: Export: Wrap Cell Content:
Activate Windows
Go to Settings to activate Windows.
coffee 1
Read Only
Desktop
12:52 PM
3/25/2022

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

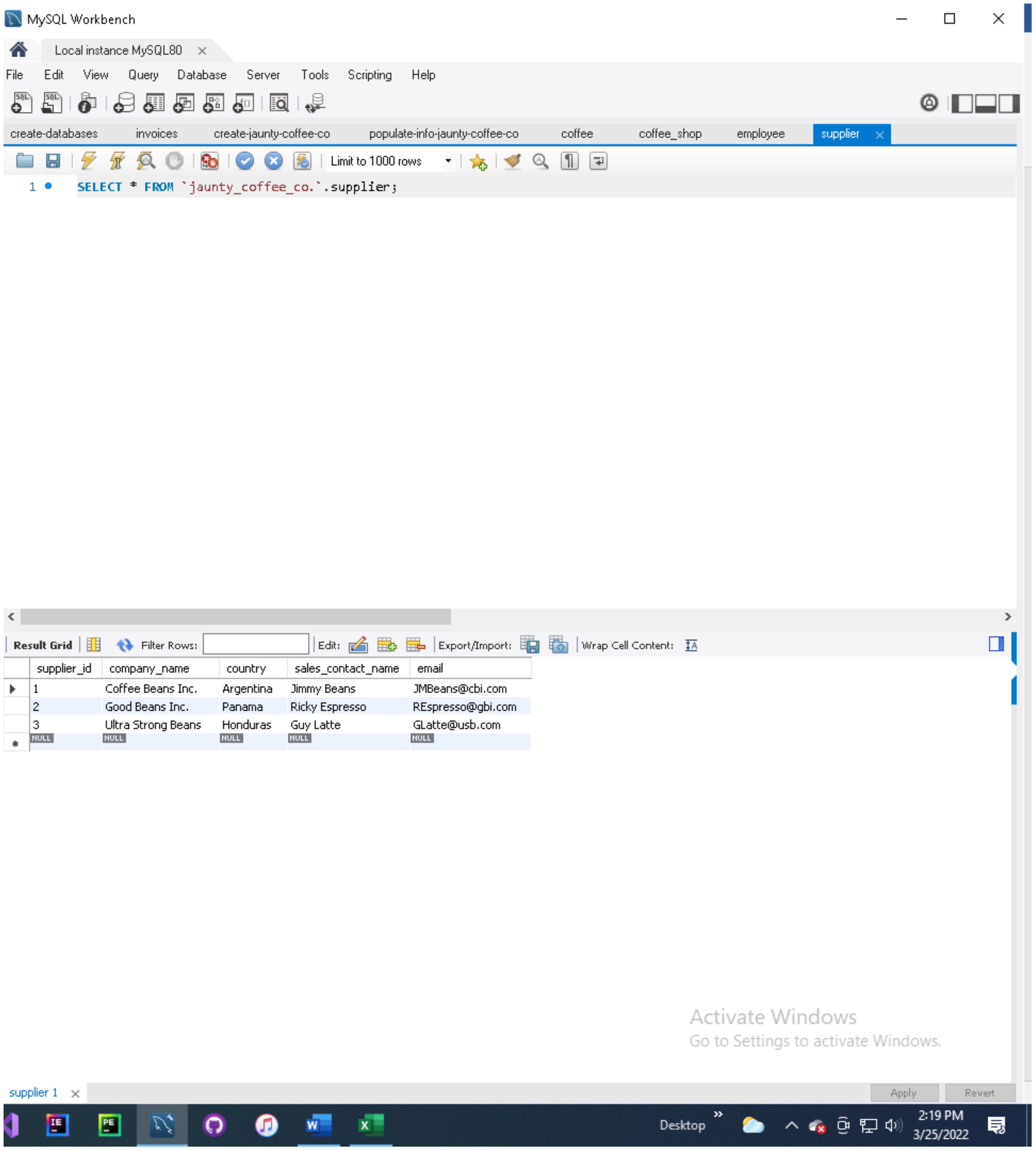

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
create-databases invoices create-jaunty-coffee-co populate-info-jaunty-coffee-co coffee coffee_shop employee supplier
Limit to 1000 rows
Activate Windows
Go to Settings to activate Windows.
Desktop
2:15 PM
3/25/2022

```sql
INSERT INTO COFFEE_SHOP (shop_id, shop_name, city, state)
VALUES
(1, "Jaunty Urban Coffee", "Richmond", "VA"),
(2, "Jaunty Old West Coffee", "Austin", "TX"),
(3, "Jaunty Vacation Coffee", "Honolu", "HI");

INSERT INTO SUPPLIER (supplier_id, company_name, country, sales_contact_name, email)
VALUES
(1, "Coffee Beans Inc.", "Argentina", "Jimmy Beans", "JMBeans@cbi.com"),
(2, "Good Beans Inc.", "Panama", "Ricky Espresso", "REspresso@gbi.com"),
(3, "Ultra Strong Beans ", "Honduras", "Guy Latte", "GLatte@usb.com");

INSERT INTO EMPLOYEE (employee_id, first_name, last_name, hire_date, job_title)
VALUES
(1, "Managa", "Hode", "2018-01-01", "Manager"),
(2, "Lazlo", "Paquito", "2020-05-01", "Marketing Lead"),
(3, "Jacob", "Hafza", "2019-03-02", "IT Lead");

INSERT INTO COFFEE (coffee_id, coffee_name, price_per_pound)
VALUES
(1, "Morning Brew", 8.25),
(2, "Hazelnut Crazy", 11.50),
(3, "Smooth Black", 14.50);

UPDATE EMPLOYEE
SET shop_id = 1 WHERE employee_id = 1;
UPDATE EMPLOYEE
SET shop_id = 2 WHERE employee_id = 2;
UPDATE EMPLOYEE
SET shop_id = 3 WHERE employee_id = 3;

UPDATE COFFEE
SET shop_id = 1, supplier_id = 1 WHERE coffee_id = 3;
UPDATE COFFEE
SET shop_id = 2, supplier_id = 2 WHERE coffee_id = 1;
UPDATE COFFEE
SET shop_id = 3, supplier_id = 3 WHERE coffee_id = 2;
```


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
create-databases invoices create-jaunty-coffee-co populate-info-jaunty-coffee-co coffee coffee_shop employee supplier
Limit to 1000 rows
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
coffee_shop 1
Apply Revert
Activate Windows
Go to Settings to activate Windows.
Desktop 2:18 PM 3/25/2022

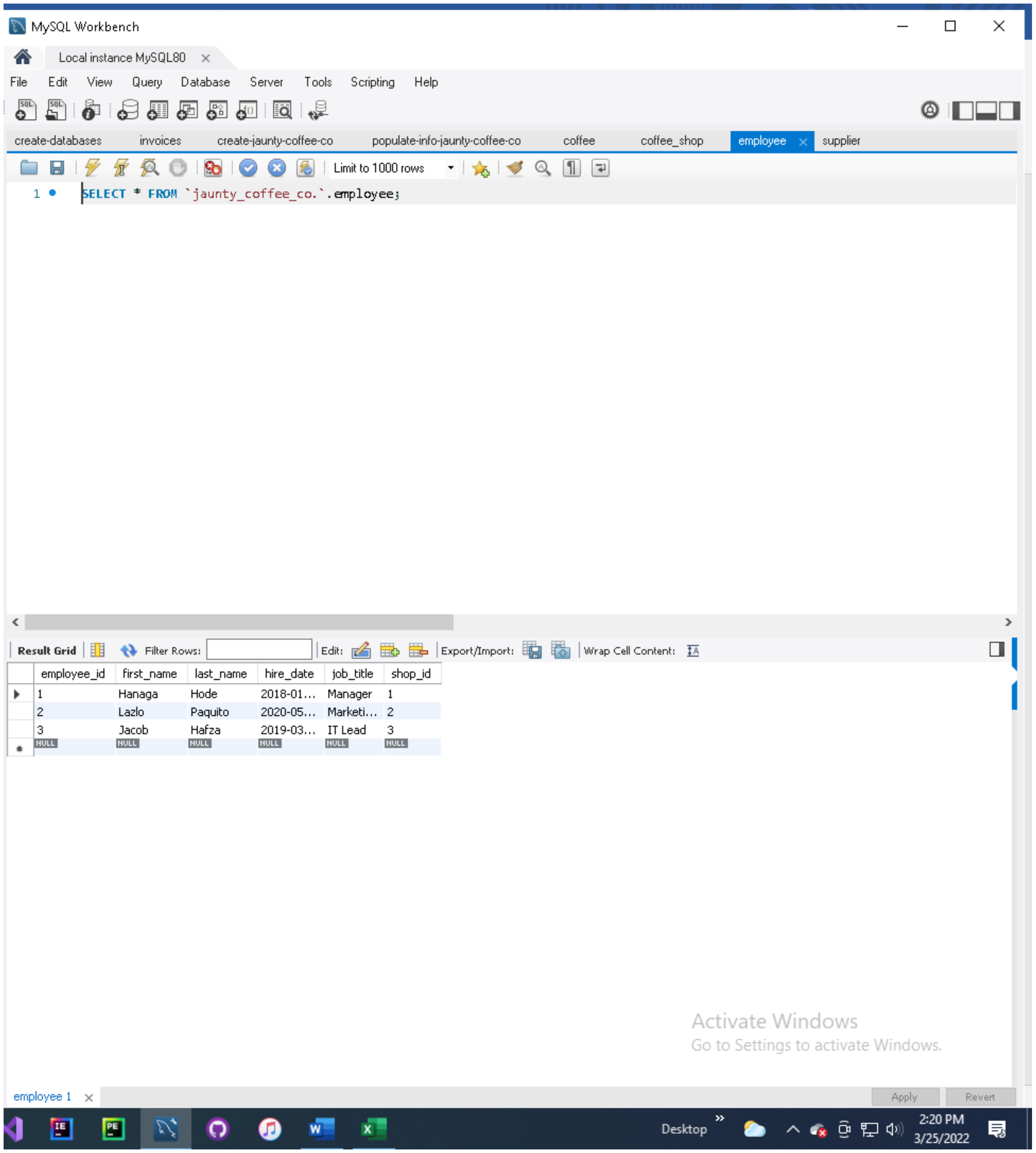
```
SELECT * FROM `jaunty_coffee_co`.coffee_shop;
```

shop_id	shop_name	city	state
1	Jaunty Urban Coffee	Richmond	VA
2	Jaunty Old West Coffee	Austin	TX
3	Jaunty Vacation Coffee	Honolu	HI
NULL	NULL	NULL	NULL


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
create-databases invoices create-jaunty-coffee-co populate-info-jaunty-coffee-co coffee coffee_shop employee supplier
Limit to 1000 rows
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
Activate Windows
Go to Settings to activate Windows.
supplier 1
Apply Revert
Desktop
2:19 PM
3/25/2022

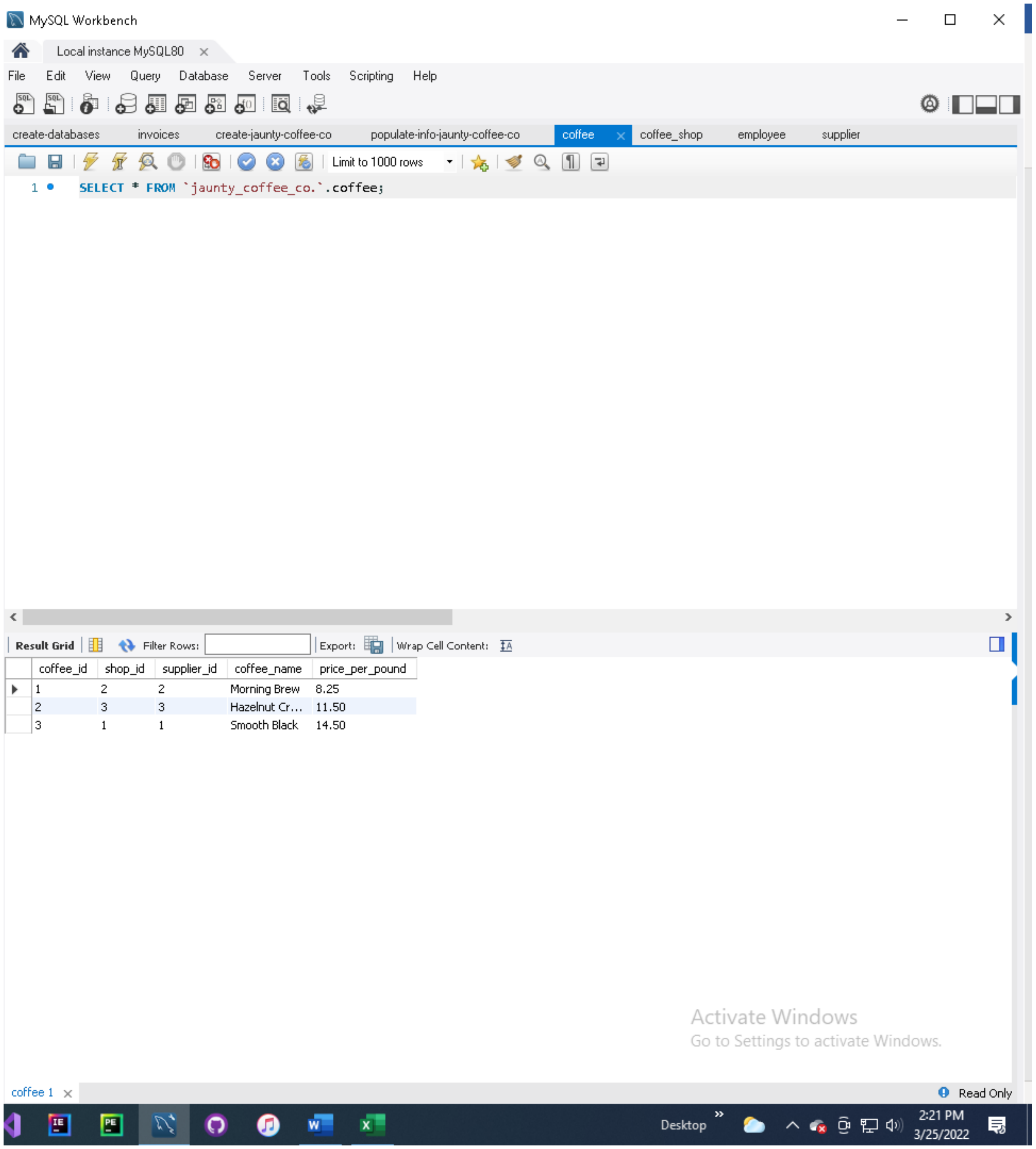
```
SELECT * FROM `jaunty_coffee_co.`.supplier;
```

supplier_id	company_name	country	sales_contact_name	email
1	Coffee Beans Inc.	Argentina	Jimmy Beans	JMBeans@cbi.com
2	Good Beans Inc.	Panama	Ricky Espresso	REspresso@gbi.com
3	Ultra Strong Beans	Honduras	Guy Latte	GLatte@usb.com
NULL	NULL	NULL	NULL	NULL


```
SELECT * FROM `jaunty_coffee_co.`.employee;
```

employee_id	first_name	last_name	hire_date	job_title	shop_id
1	Hanaga	Hode	2018-01...	Manager	1
2	Lazlo	Paquito	2020-05...	Marketi...	2
3	Jacob	Hafza	2019-03...	IT Lead	3
NULL	NULL	NULL	NULL	NULL	NULL

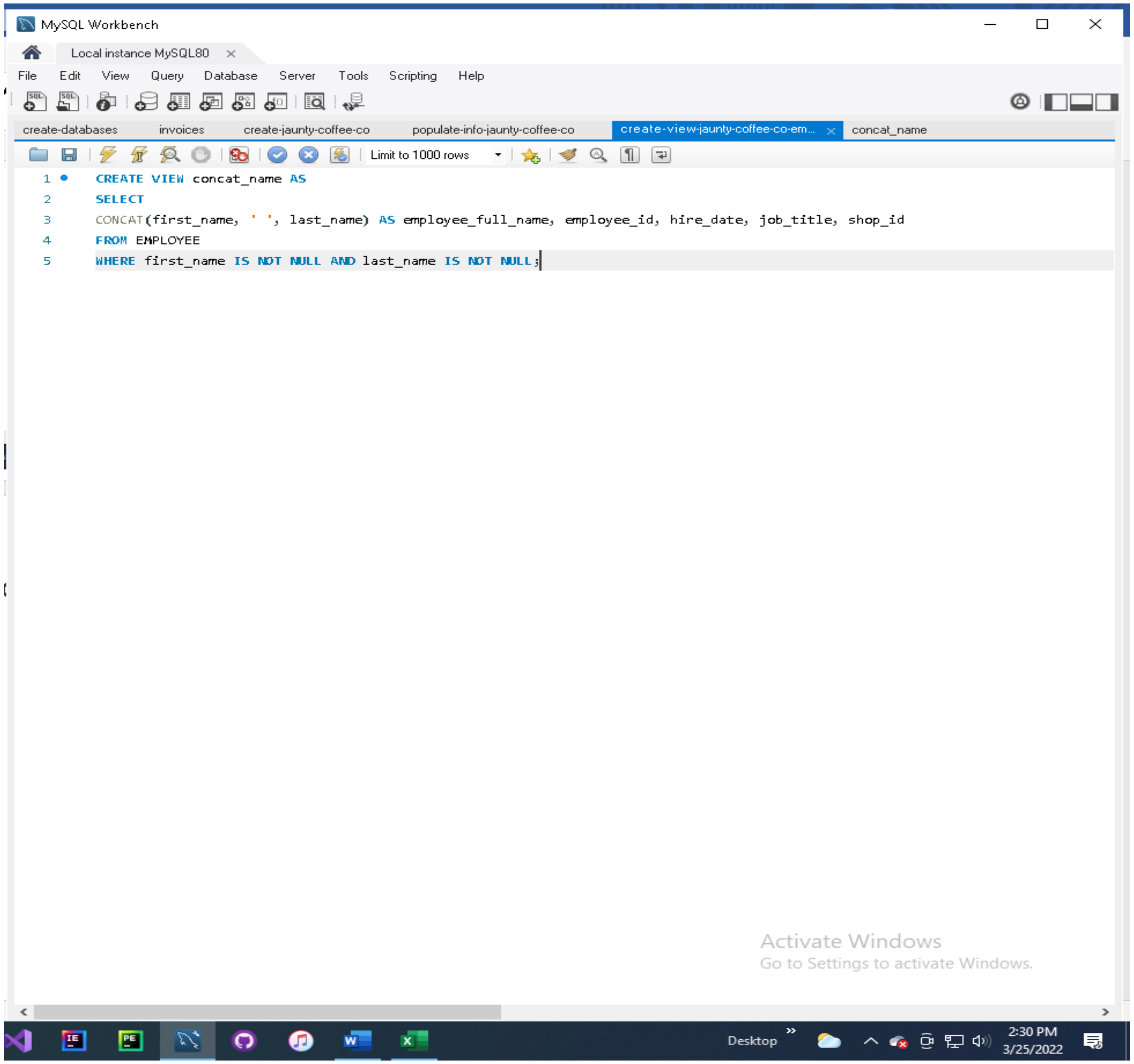
MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
create-databases invoices create-jaunty-coffee-co populate-info-jaunty-coffee-co coffee coffee_shop employee supplier
Limit to 1000 rows
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
Activate Windows
Go to Settings to activate Windows.
employee 1
Apply Revert
Desktop
2:20 PM
3/25/2022



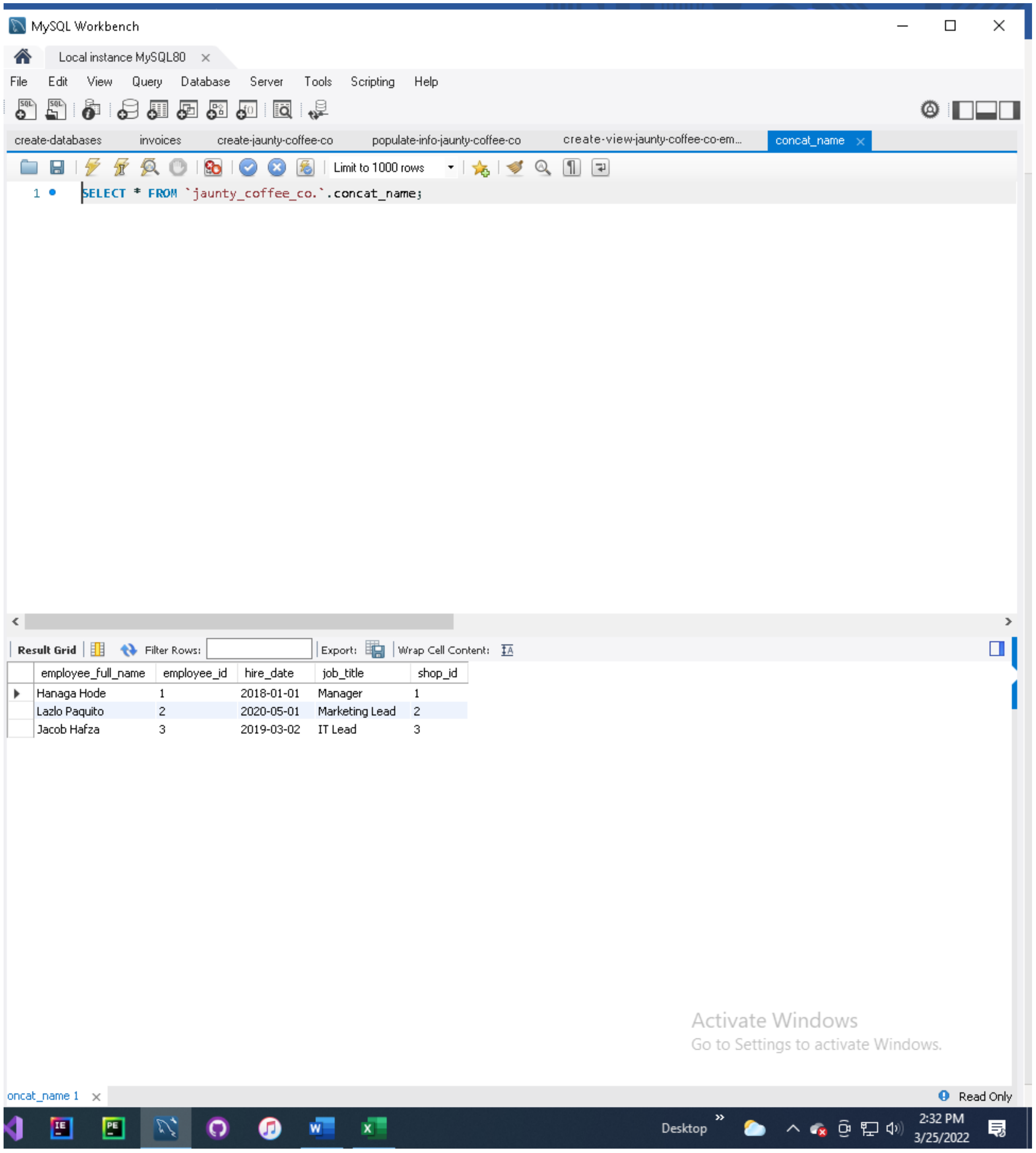
```
SELECT * FROM `jaunty_coffee_co.`.coffee;
```

coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
1	2	2	Morning Brew	8.25
2	3	3	Hazelnut Cr...	11.50
3	1	1	Smooth Black	14.50

3. Develop SQL code to create a view from EMPLOYEE table, with new employee_full_name attribute



```sql
CREATE VIEW concat_name AS
SELECT
CONCAT(first_name, ' ', last_name) AS employee_full_name, employee_id, hire_date, job_title, shop_id
FROM EMPLOYEE
WHERE first_name IS NOT NULL AND last_name IS NOT NULL;
```

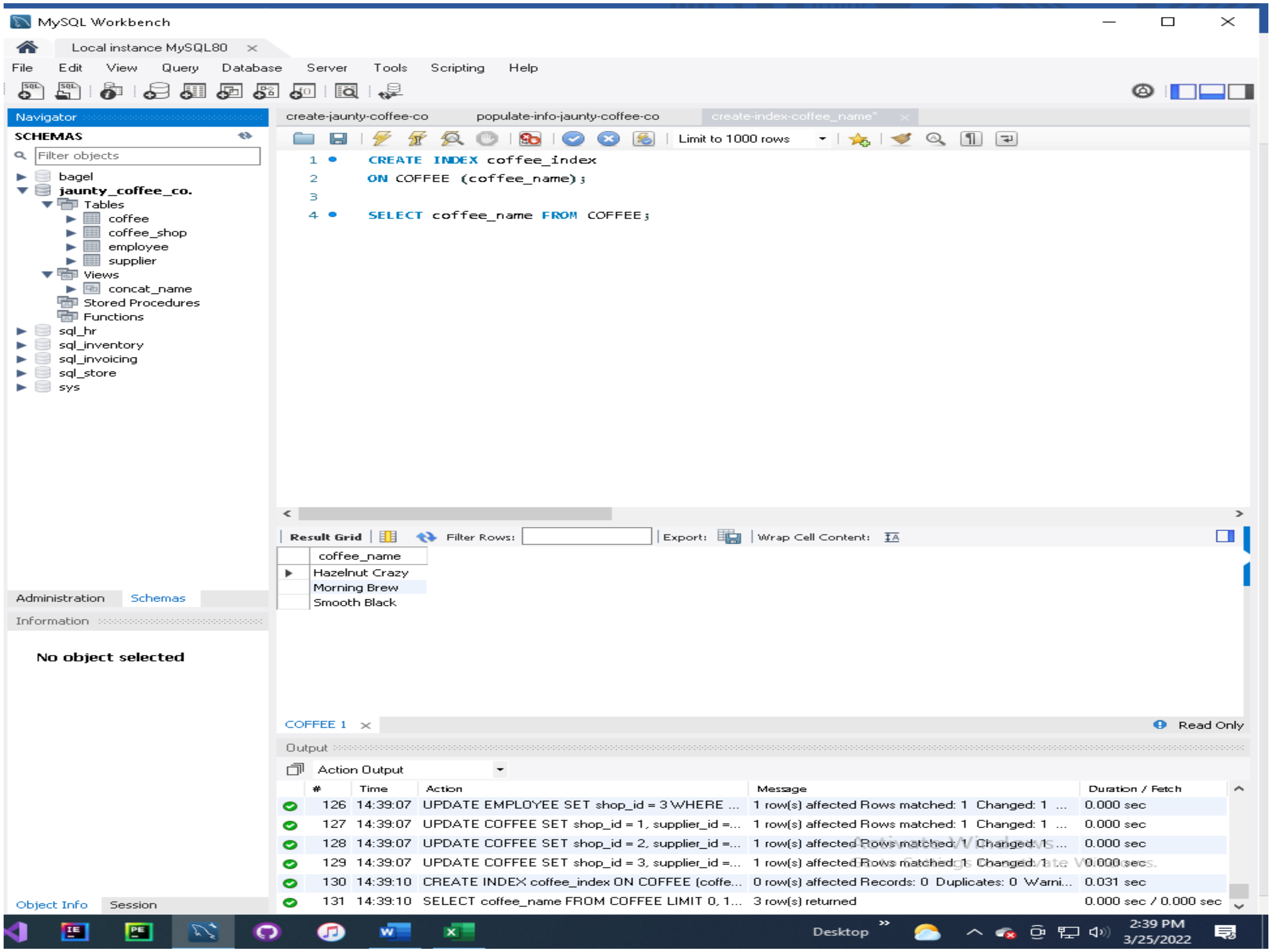


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
create-databases invoices create-jaunty-coffee-co populate-info-jaunty-coffee-co create-view-jaunty-coffee-co-em... concat_name
Limit to 1000 rows
Result Grid Filter Rows: Export: Wrap Cell Content:
oncat_name 1
Read Only
Desktop
2:32 PM
3/25/2022

```sql
SELECT * FROM `jaunty_coffee_co.`.concat_name;
```

employee_full_name	employee_id	hire_date	job_title	shop_id
Hanaga Hode	1	2018-01-01	Manager	1
Lazlo Paquito	2	2020-05-01	Marketing Lead	2
Jacob Hafza	3	2019-03-02	IT Lead	3

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

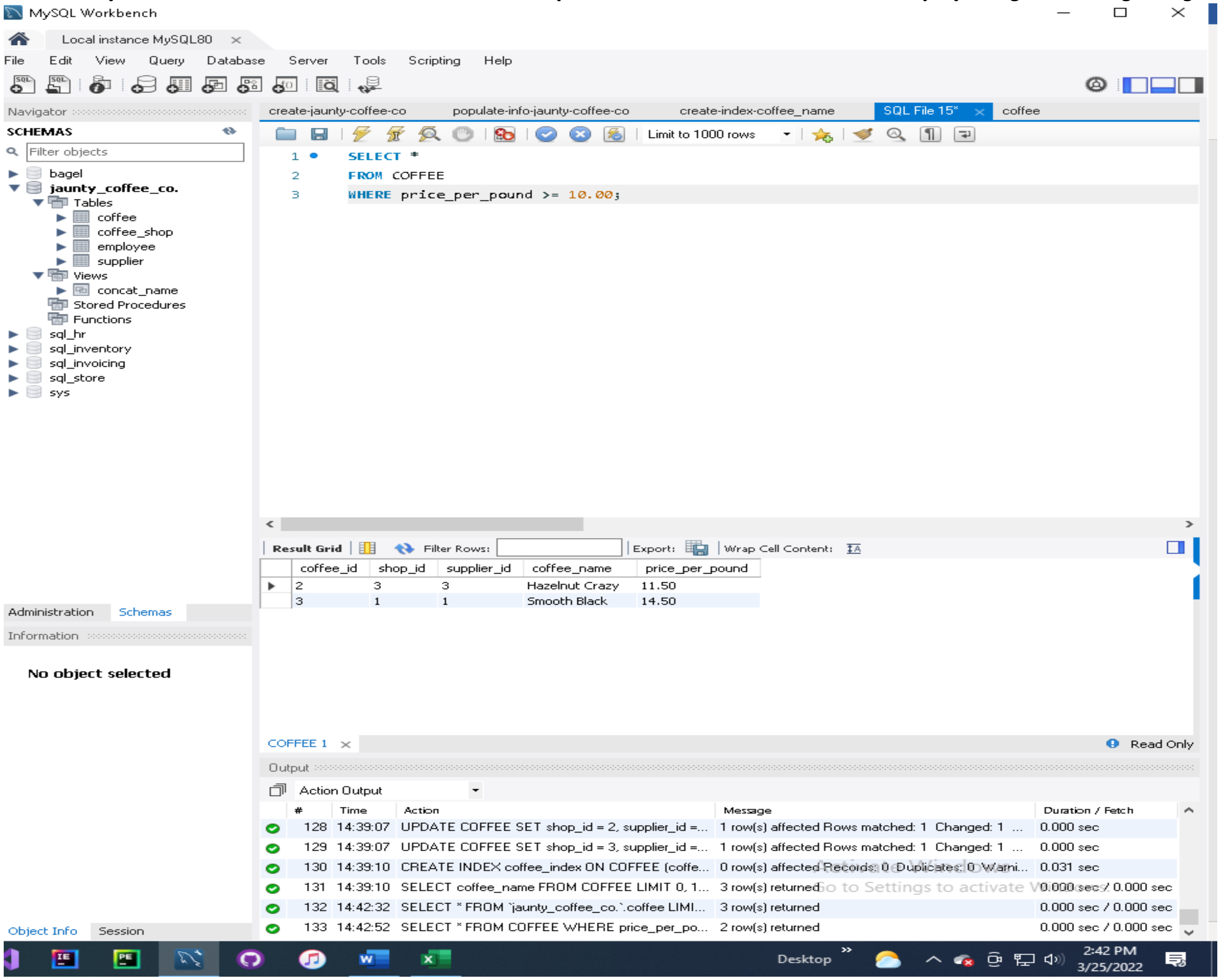


```sql
CREATE INDEX coffee_index
ON COFFEE (coffee_name);

SELECT coffee_name FROM COFFEE;
```

coffee_name
Hazelnut Crazy
Morning Brew
Smooth Black

```
#    Time      Action                                         Message                                              Duration / Fetch
126  14:39:07  UPDATE EMPLOYEE SET shop_id = 3 WHERE ...      1 row(s) affected Rows matched: 1  Changed: 1  ...  0.000 sec
127  14:39:07  UPDATE COFFEE SET shop_id = 1, supplier_id =...  1 row(s) affected Rows matched: 1  Changed: 1  ...  0.000 sec
128  14:39:07  UPDATE COFFEE SET shop_id = 2, supplier_id =...  1 row(s) affected Rows matched: 1  Changed: 1  ...  0.000 sec
129  14:39:07  UPDATE COFFEE SET shop_id = 3, supplier_id =...  1 row(s) affected Rows matched: 1  Changed: 1  ...  0.000 sec
130  14:39:10  CREATE INDEX coffee_index ON COFFEE (coffe...   0 row(s) affected Records: 0  Duplicates: 0  Warni...  0.031 sec
131  14:39:10  SELECT coffee_name FROM COFFEE LIMIT 0, 1...   3 row(s) returned                                    0.000 sec / 0.000 sec
```

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

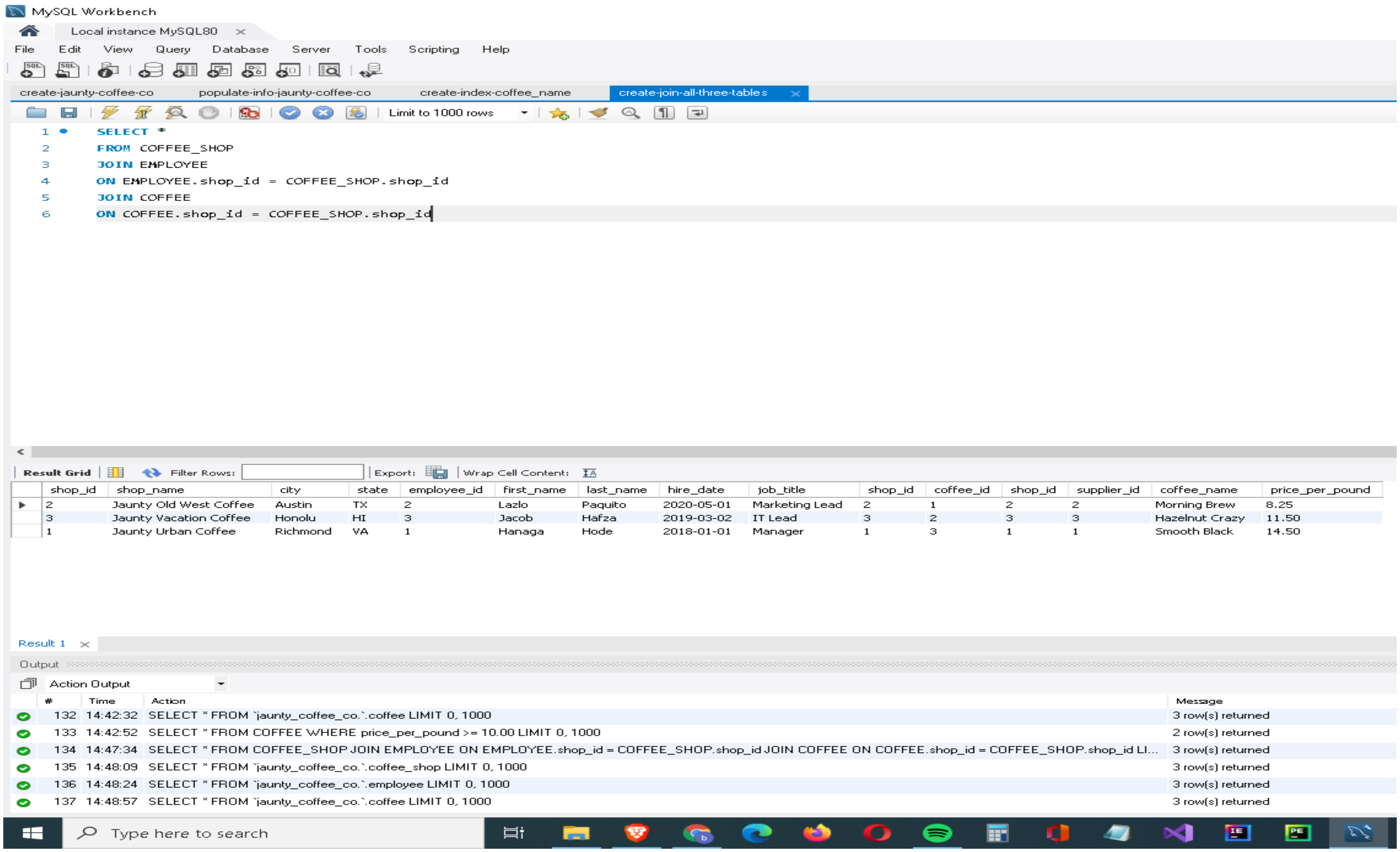


```sql
SELECT *
FROM COFFEE
WHERE price_per_pound >= 10.00;
```

coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
2	3	3	Hazelnut Crazy	11.50
3	1	1	Smooth Black	14.50

```
#    Time      Action                                        Message                                             Duration / Fetch
128  14:39:07  UPDATE COFFEE SET shop_id = 2, supplier_id =...  1 row(s) affected Rows matched: 1  Changed: 1  ...  0.000 sec
129  14:39:07  UPDATE COFFEE SET shop_id = 3, supplier_id =...  1 row(s) affected Rows matched: 1  Changed: 1  ...  0.000 sec
130  14:39:10  CREATE INDEX coffee_index ON COFFEE (coffe...     0 row(s) affected Records: 0  Duplicates: 0  Warni...  0.031 sec
131  14:39:10  SELECT coffee_name FROM COFFEE LIMIT 0, 1...      3 row(s) returned                                   0.000 sec / 0.000 sec
132  14:42:32  SELECT * FROM `jaunty_coffee_co.`.coffee LIMI...  3 row(s) returned                                   0.000 sec / 0.000 sec
133  14:42:52  SELECT * FROM COFFEE WHERE price_per_po...        2 row(s) returned                                   0.000 sec / 0.000 sec
```

6. Develop SQL code to create a query by joining three different tables, including attributes from all three tables



```sql
SELECT *
FROM COFFEE_SHOP
JOIN EMPLOYEE
ON EMPLOYEE.shop_id = COFFEE_SHOP.shop_id
JOIN COFFEE
ON COFFEE.shop_id = COFFEE_SHOP.shop_id
```

shop_id	shop_name	city	state	employee_id	first_name	last_name	hire_date	job_title	shop_id	coffee_id	shop_id	supplier_id	coffee_name	price_per_pound
2	Jaunty Old West Coffee	Austin	TX	2	Lazlo	Paquito	2020-05-01	Marketing Lead	2	1	2	2	Morning Brew	8.25
3	Jaunty Vacation Coffee	Honolu	HI	3	Jacob	Hafza	2019-03-02	IT Lead	3	2	3	3	Hazelnut Crazy	11.50
1	Jaunty Urban Coffee	Richmond	VA	1	Hanaga	Hode	2018-01-01	Manager	1	3	1	1	Smooth Black	14.50

#	Time	Action	Message
132	14:42:32	SELECT * FROM `jaunty_coffee_co`.`coffee LIMIT 0, 1000	3 row(s) returned
133	14:42:52	SELECT * FROM COFFEE WHERE price_per_pound >= 10.00 LIMIT 0, 1000	2 row(s) returned
134	14:47:34	SELECT * FROM COFFEE_SHOP JOIN EMPLOYEE ON EMPLOYEE.shop_id = COFFEE_SHOP.shop_id JOIN COFFEE ON COFFEE.shop_id = COFFEE_SHOP.shop_id LI...	3 row(s) returned
135	14:48:09	SELECT * FROM `jaunty_coffee_co`.`coffee_shop LIMIT 0, 1000	3 row(s) returned
136	14:48:24	SELECT * FROM `jaunty_coffee_co`.`employee LIMIT 0, 1000	3 row(s) returned
137	14:48:57	SELECT * FROM `jaunty_coffee_co`.`coffee LIMIT 0, 1000	3 row(s) returned