Umm AL-Qura Uniersity

Computing College at ALQunfudah

Department of Computer Science

Database Systems CS3132

1 st Semester 1446H

Advanced Database
Bakery Database Project

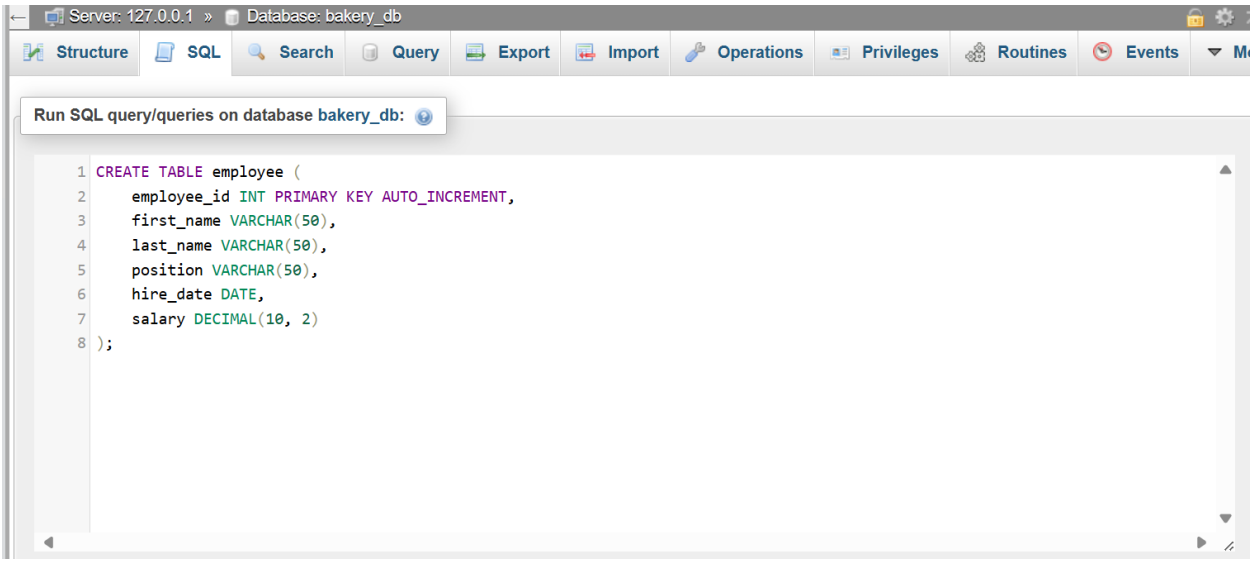
Group Members

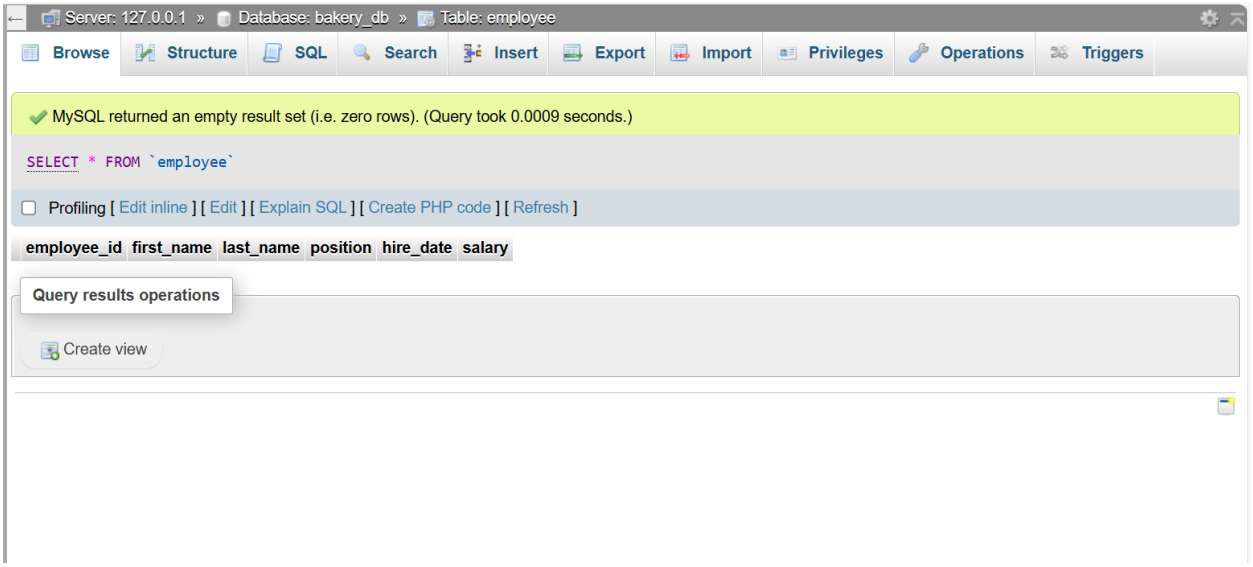
Name	ID	Group	Role
Kholod Madini Alamri	444001301	7	Crete Order Invoice Table, Insert Data to this Table, Create Procedure and call it.
Wejdan Attia AlZahrani	444004760	5	Crete Sales Table, Insert Data to this Table, Create and use transaction.
Halimah Abutaleb Alsayed	444005535	3	Crete Employee Table, Insert Data to this Table, Create index and function .
Raghad Hassan AL-Masari	444001447	7	Create Products table, Insert data to the table, Create a trigger and test the trigger.

Index and Function

1. Creating Tables

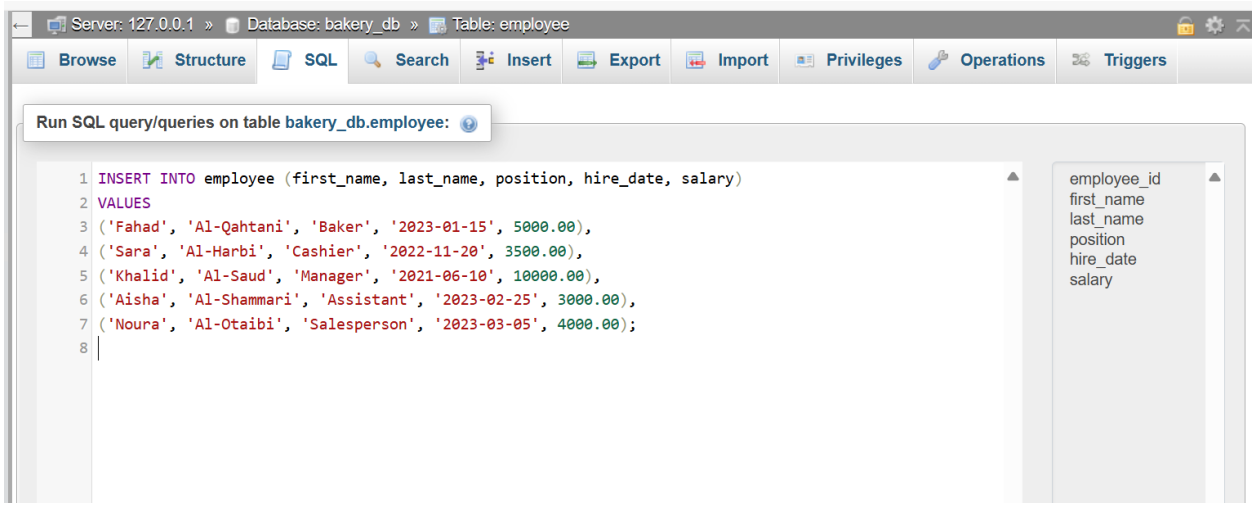
1.2 Create Employee Table

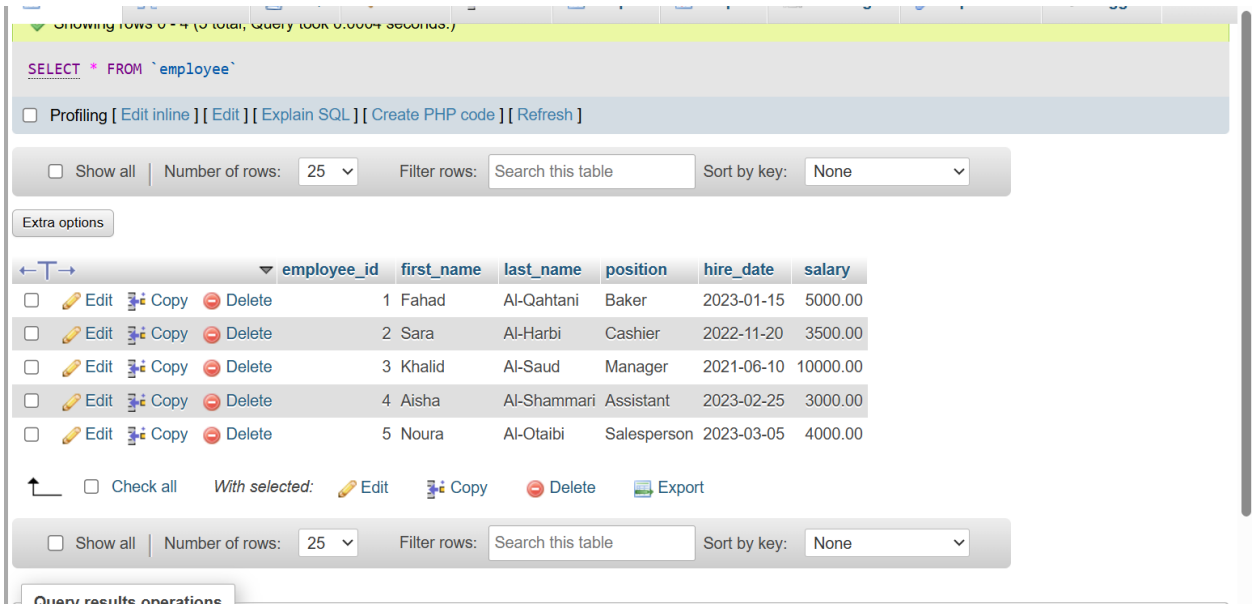




2. Inserting Data

2.2 Insert Data into Employee Table

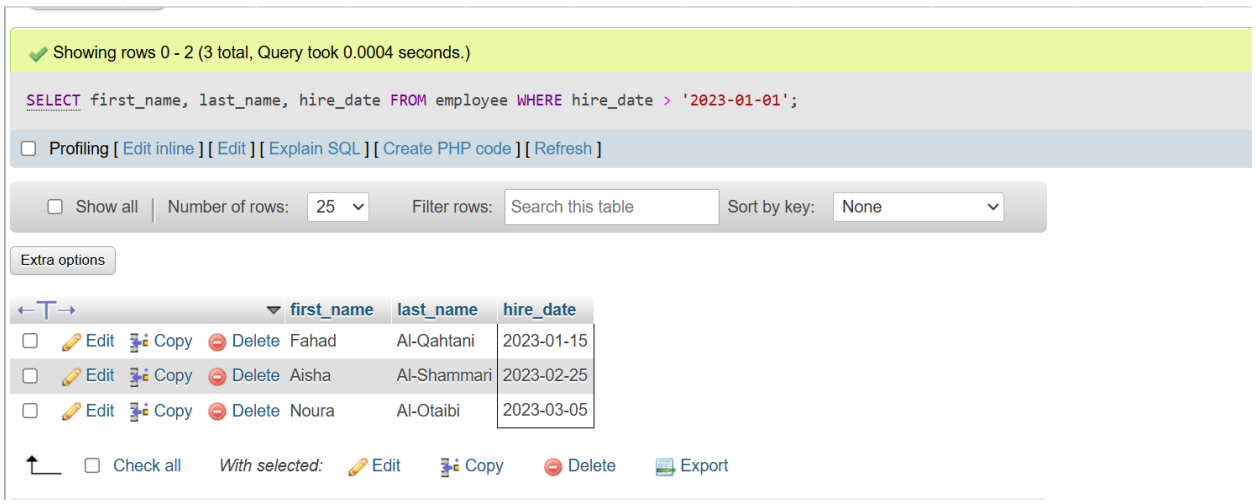




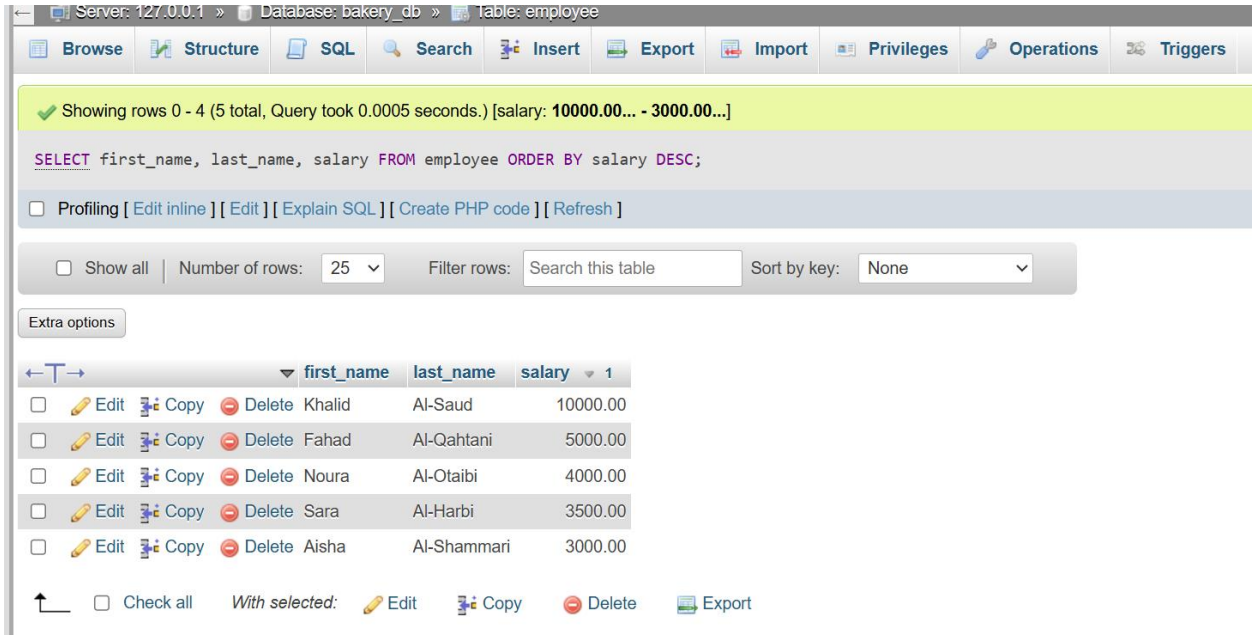
3. Select the table and select where

3.2 Select From Employee Table

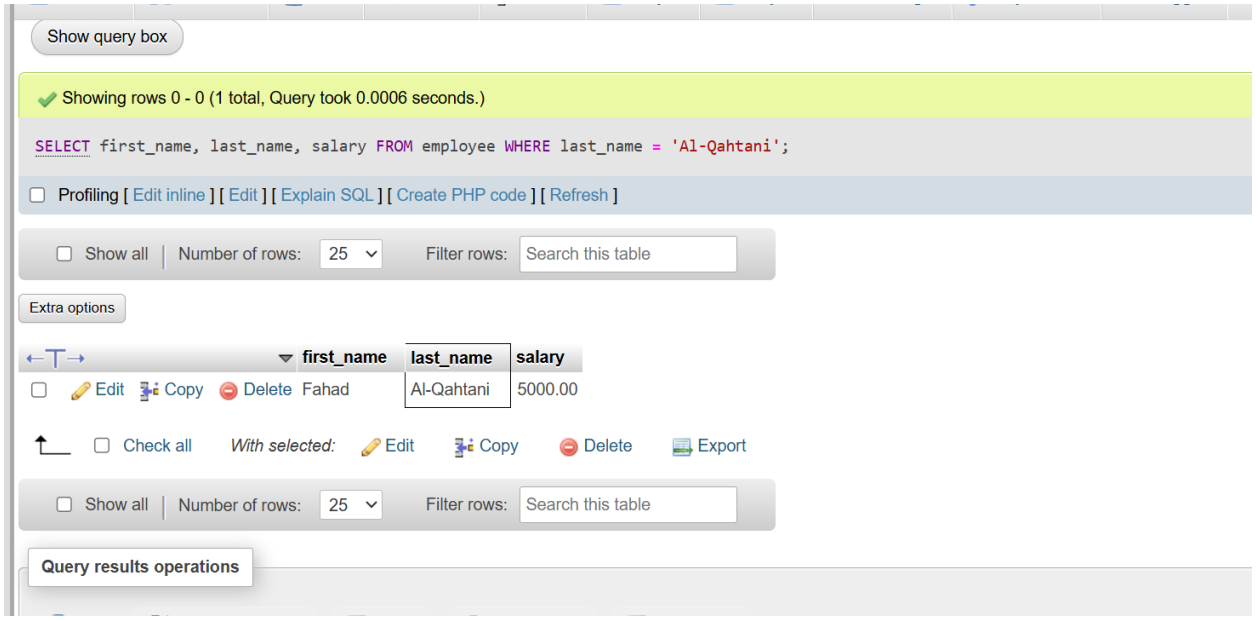
Display the first and last names and dates of employment of employees hired after this date.



Display the columns first_name, last_name, and salary from the employee table, where the results are sorted by the salary column in descending order (DESC).

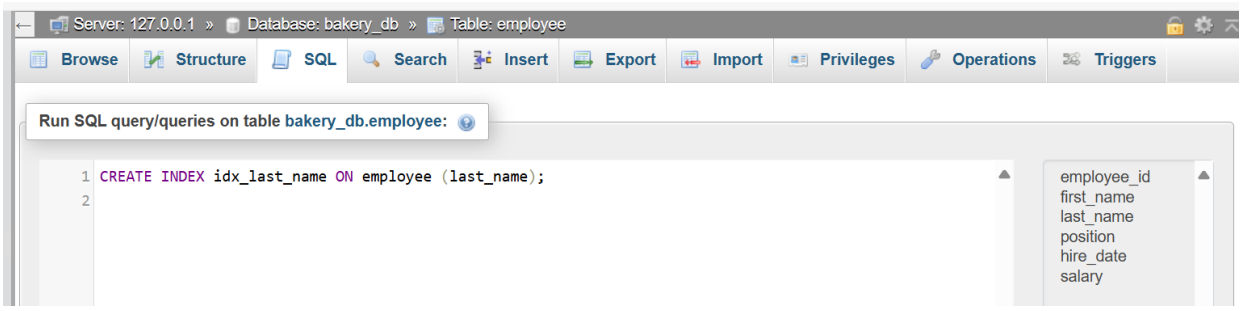


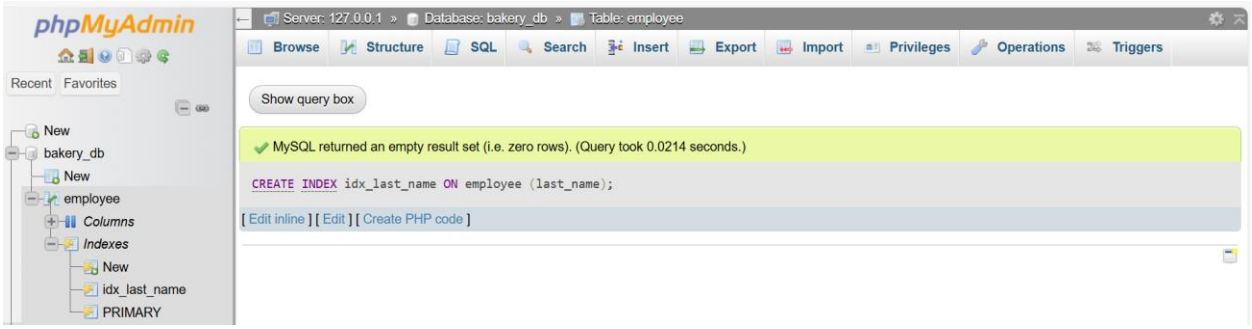
Display the first_name, last_name, and salary columns from the employee table, filtering the results based on the value in the last_name column so that it is equal to "Al-Qahtani".

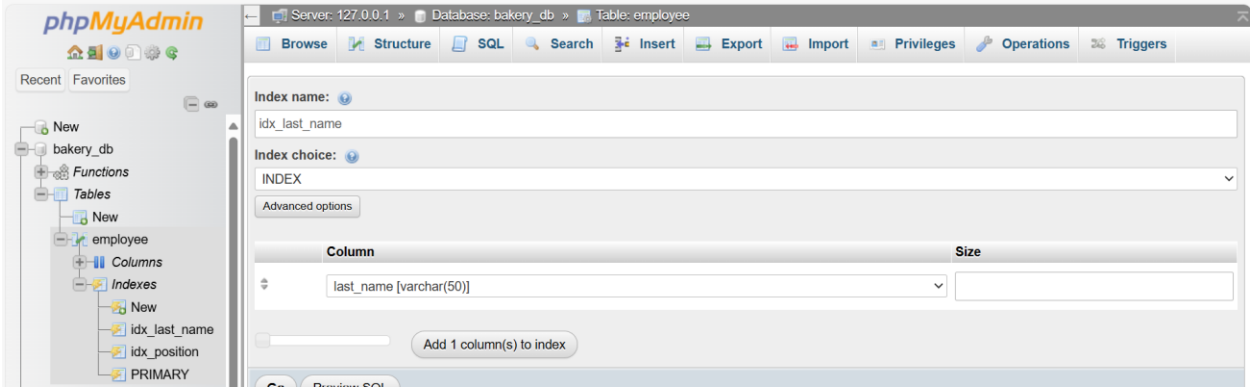


4. Create an index and function
   4.1. Create Index

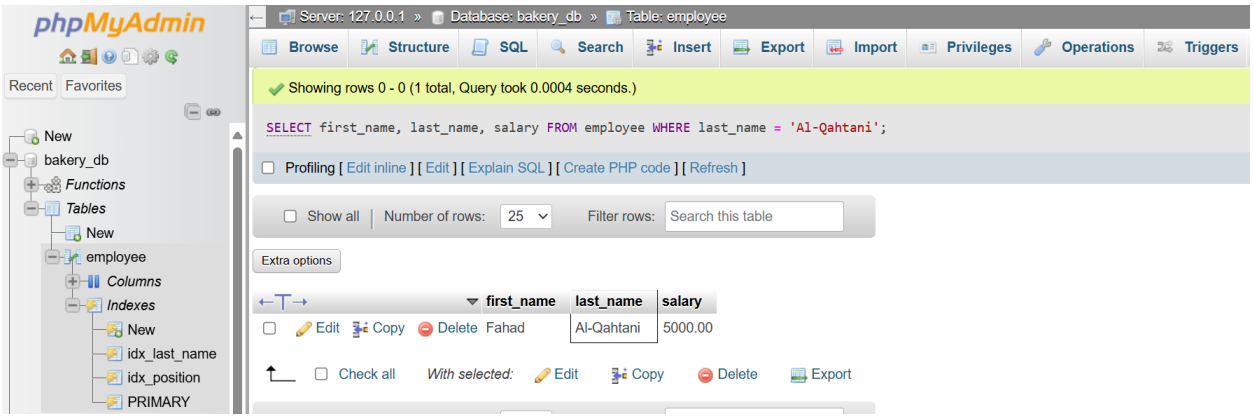
index on the employee's last_name :



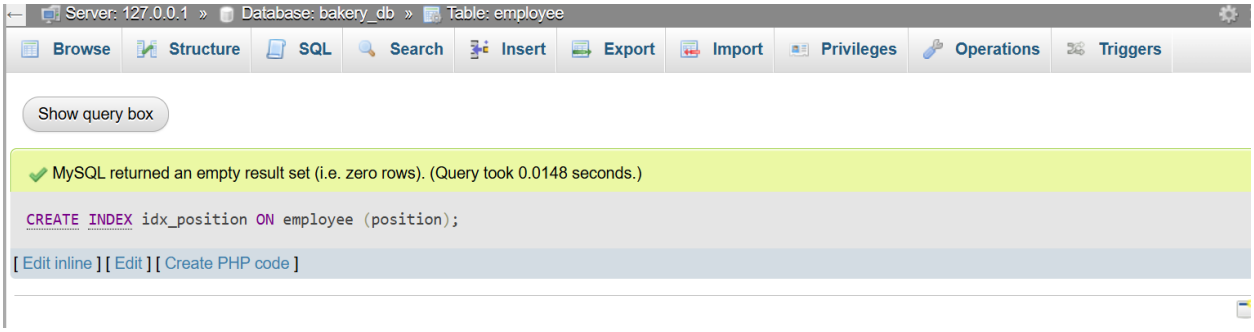


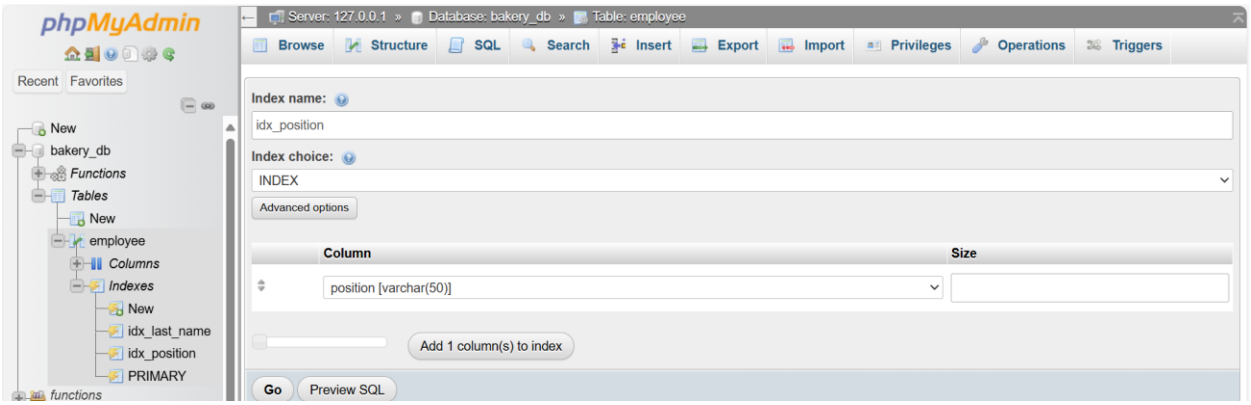


Query, idx_last_name index will help improve search speed in last_name column

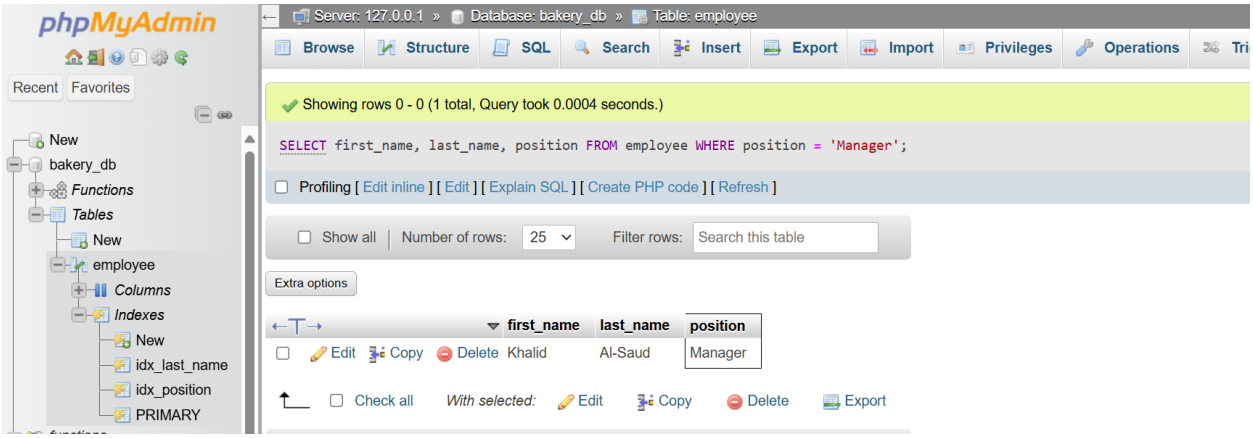


Create an Index on employee position



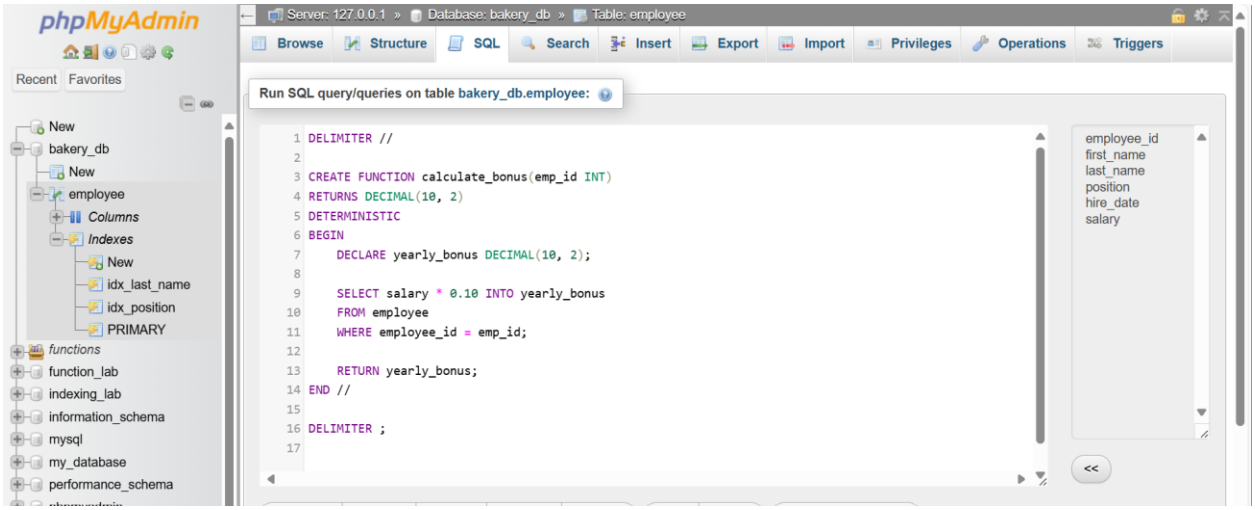


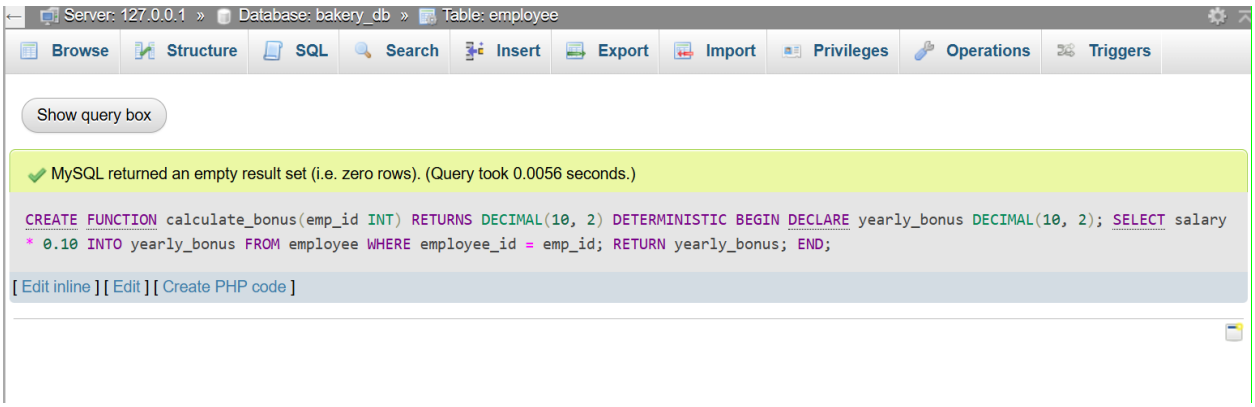
The index created on the position column will be used when we filter the results based on the employee's job position.

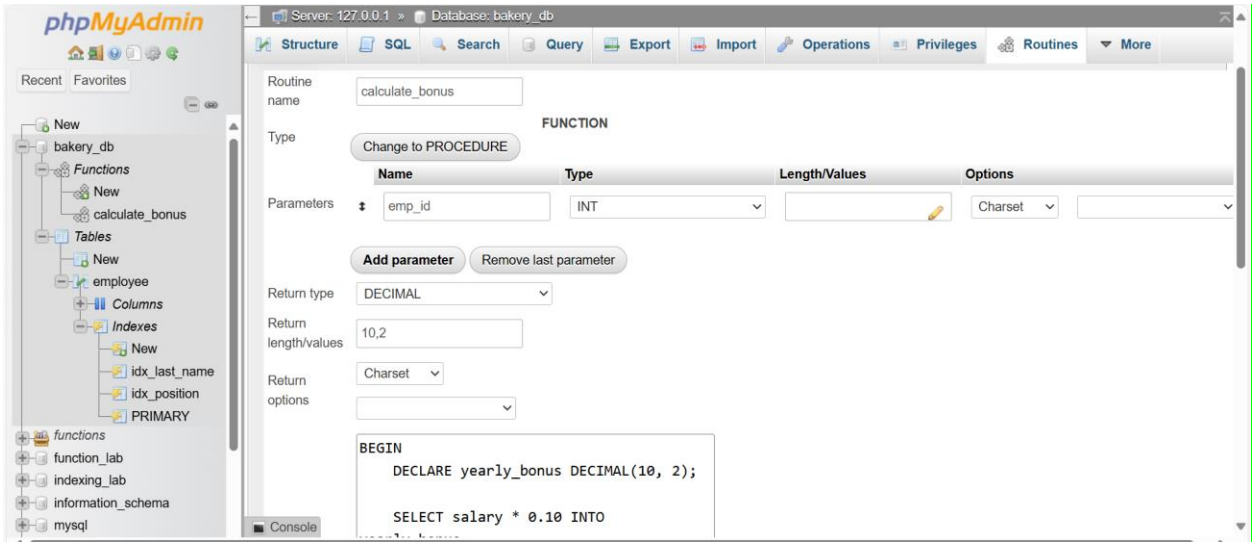


4.2. Create Function

This function calculates an annual bonus equal to 10% of a specific employee's salary based on the employee's identification number (emp_id). RETURN yearly_bonus returns the calculated annual bonus value.

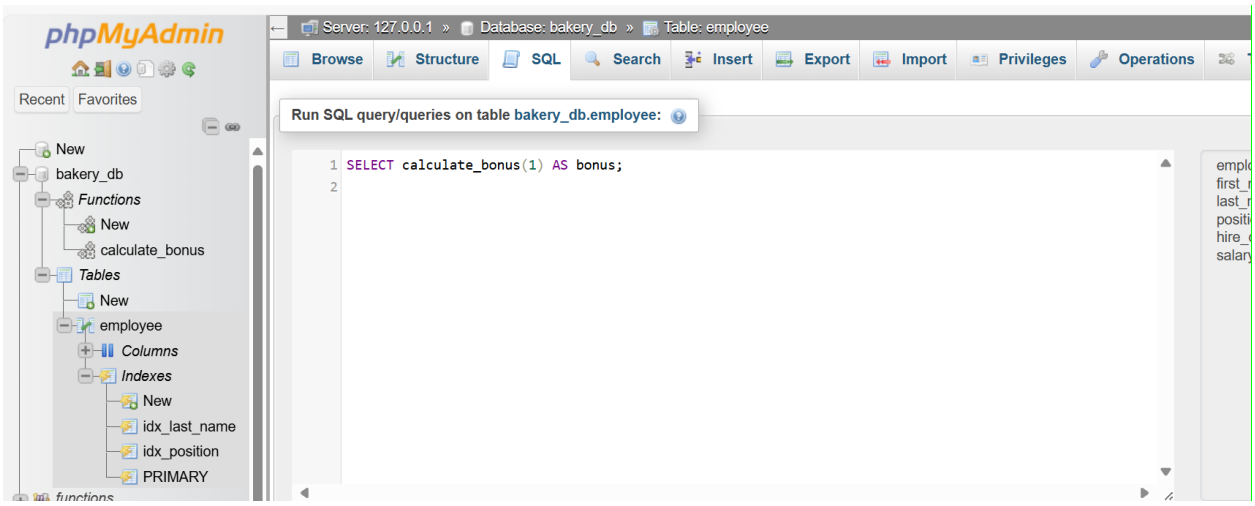


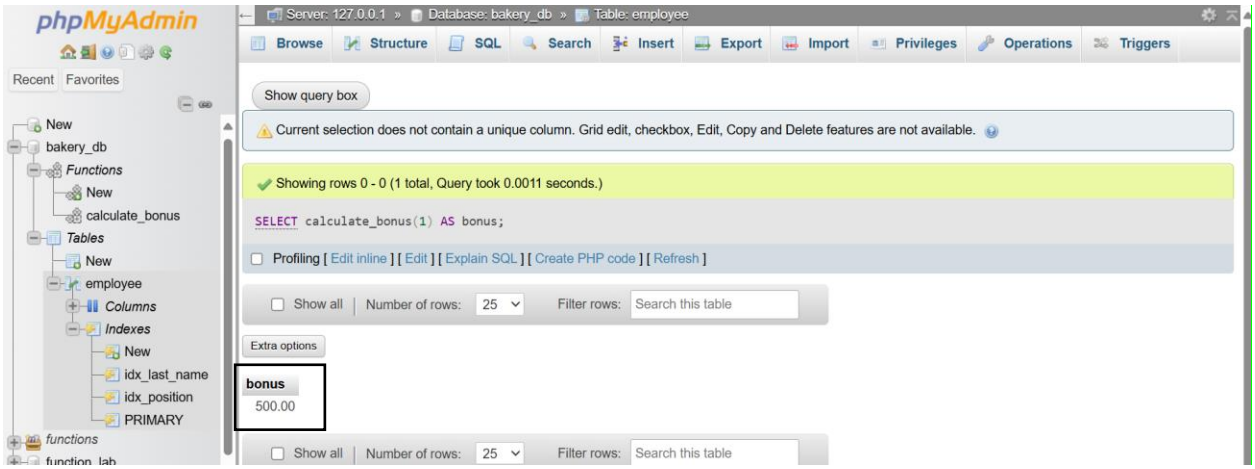




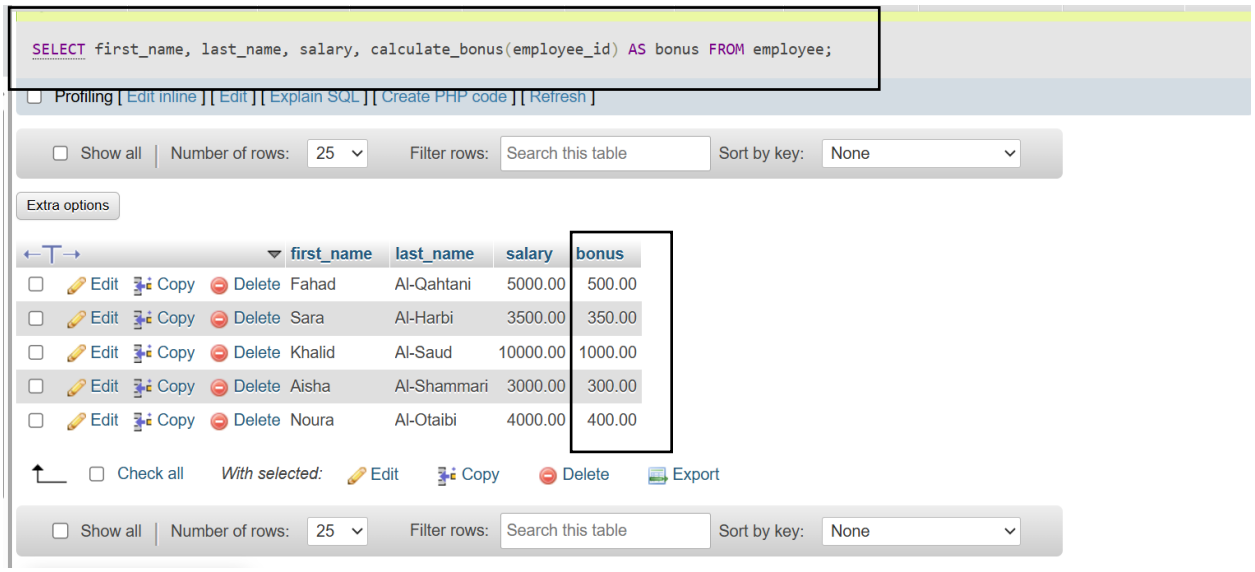
5. Call this Function:

You pass employee_id (1) to the calculate_bonus function, and the function will return the bonus for this employee. The result will be the bonus column containing the calculated bonus.



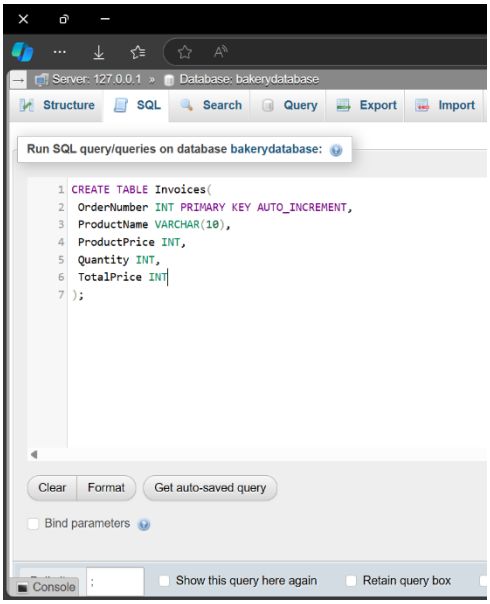


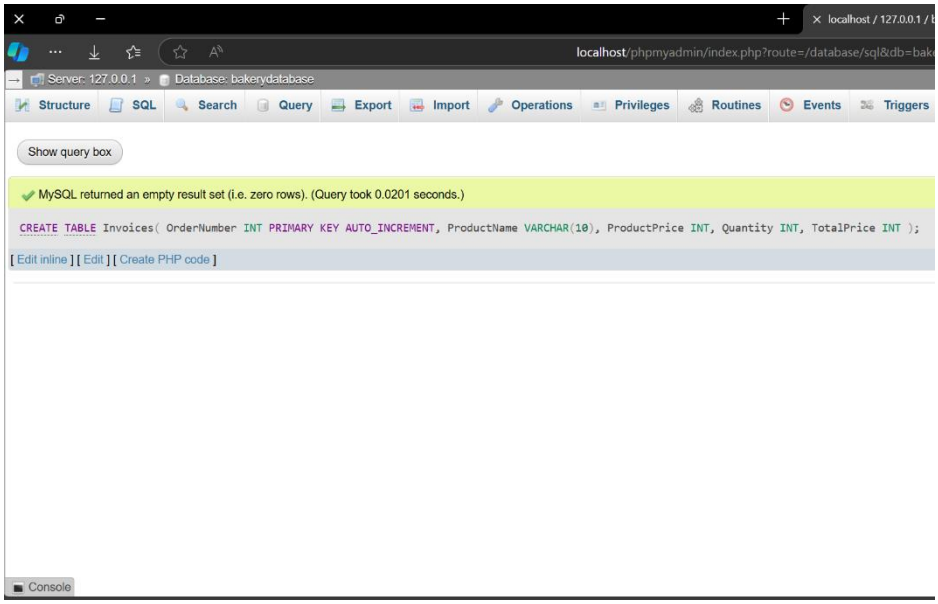
Call function for all employees To get the reward for each employee in the table, you can pass employee_id for all rows.



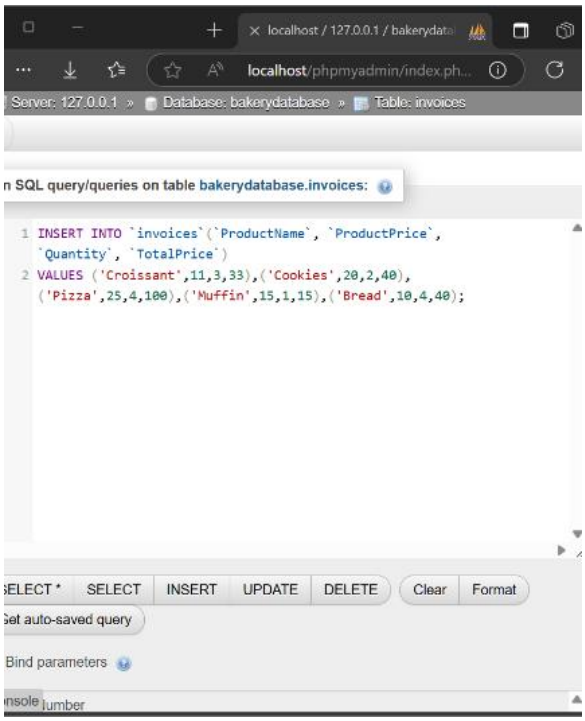
Create a Procedure and Call it

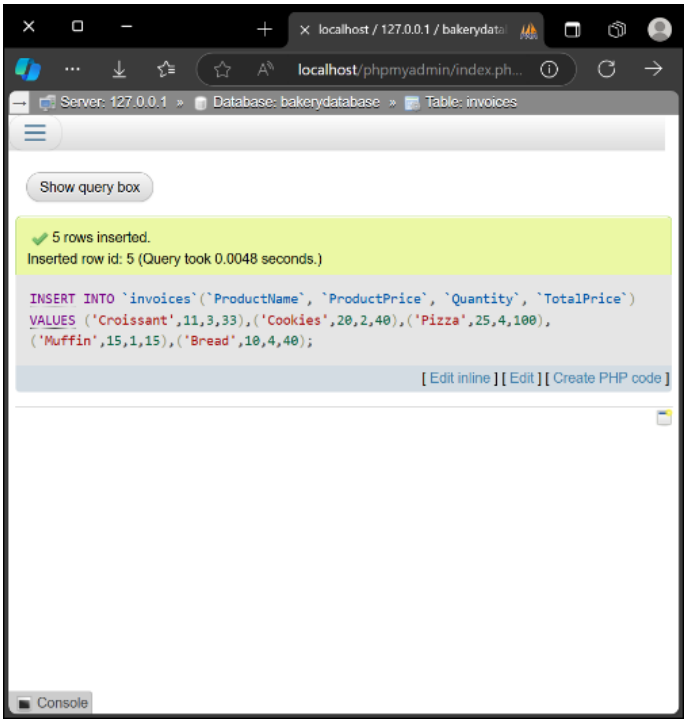
1. Create table
    1.1. Create Order Invoice table



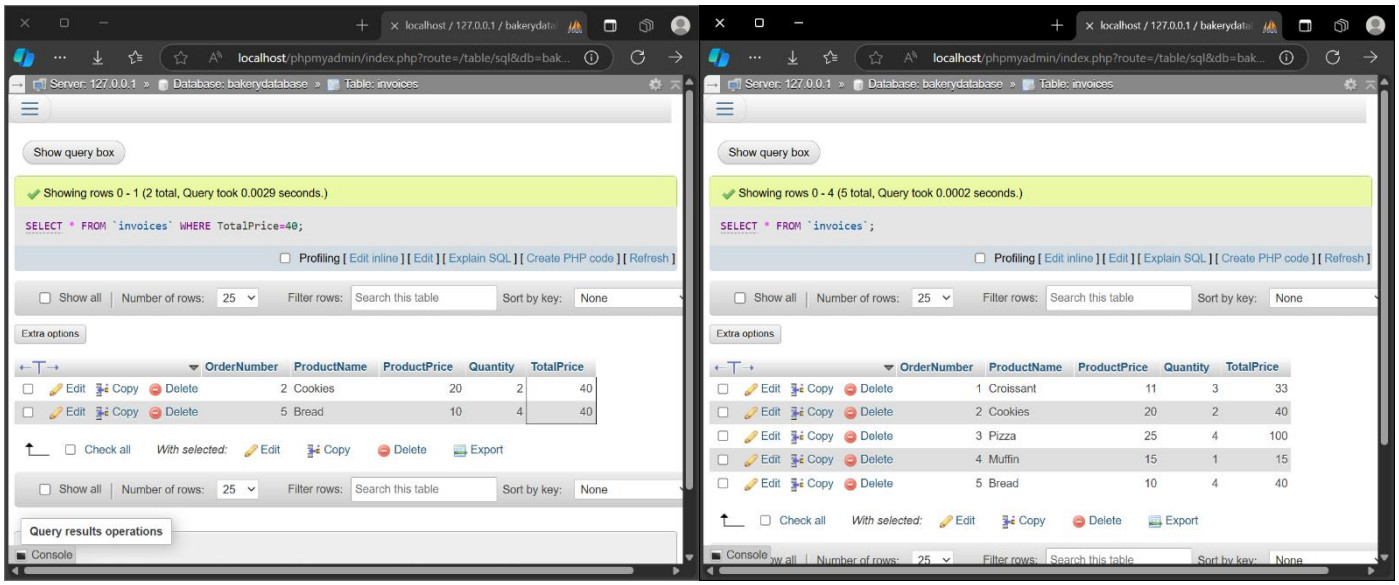


2. Insert data into table
    2.1. Insert data into Order Invoice table

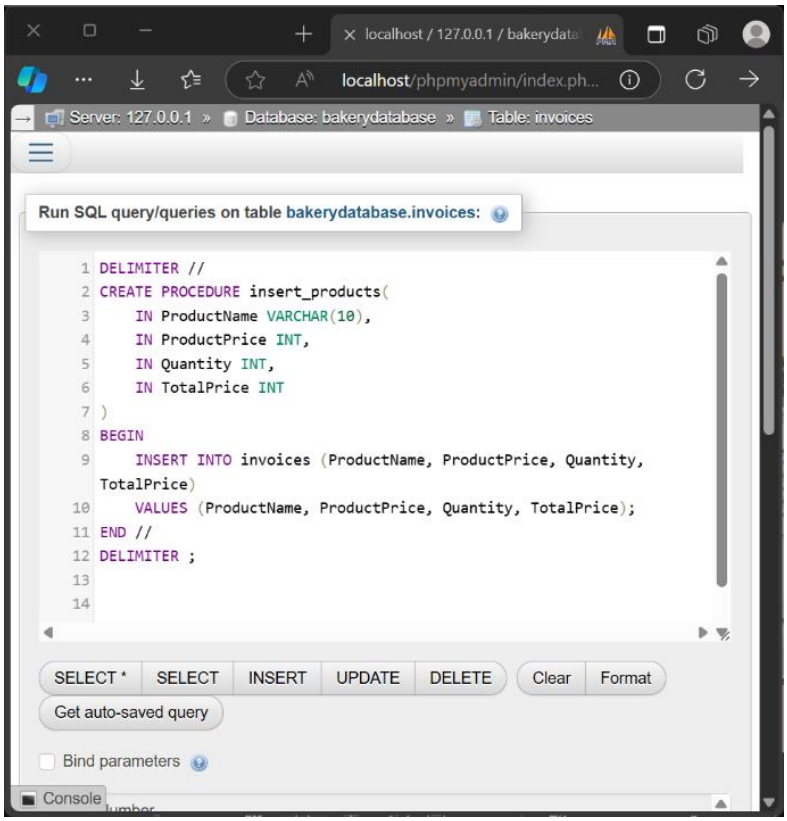




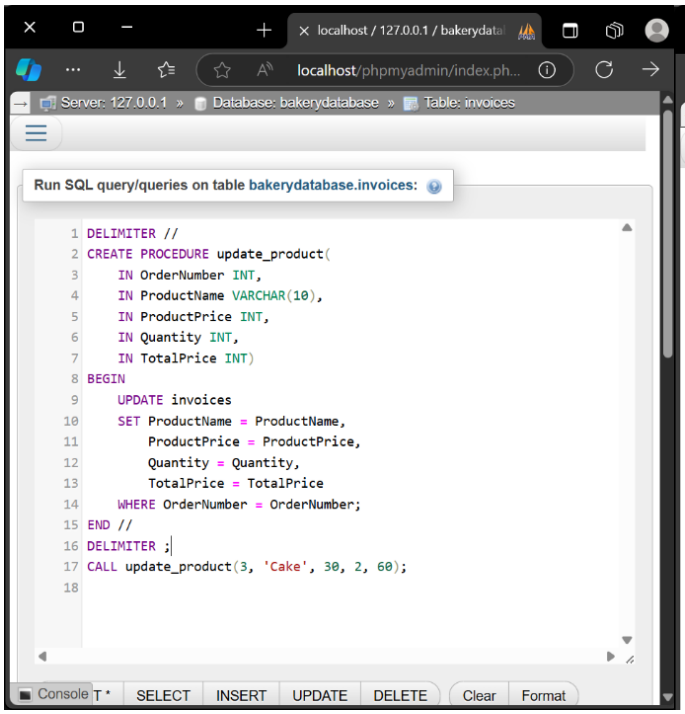
3. Select and Select where

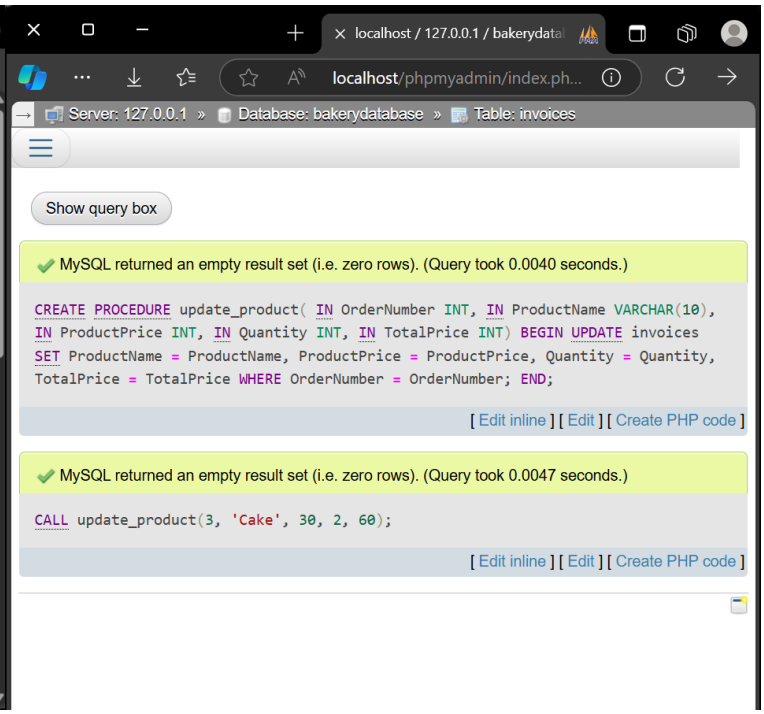


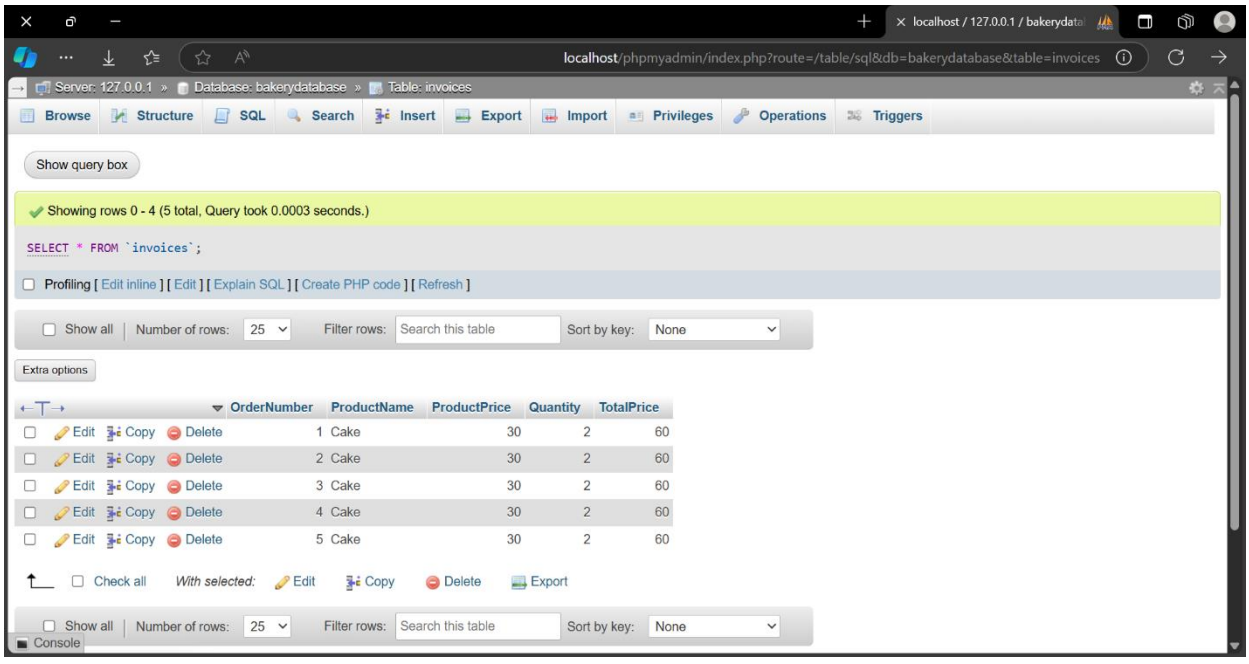
4. Create a Stored Procedure for Insertion



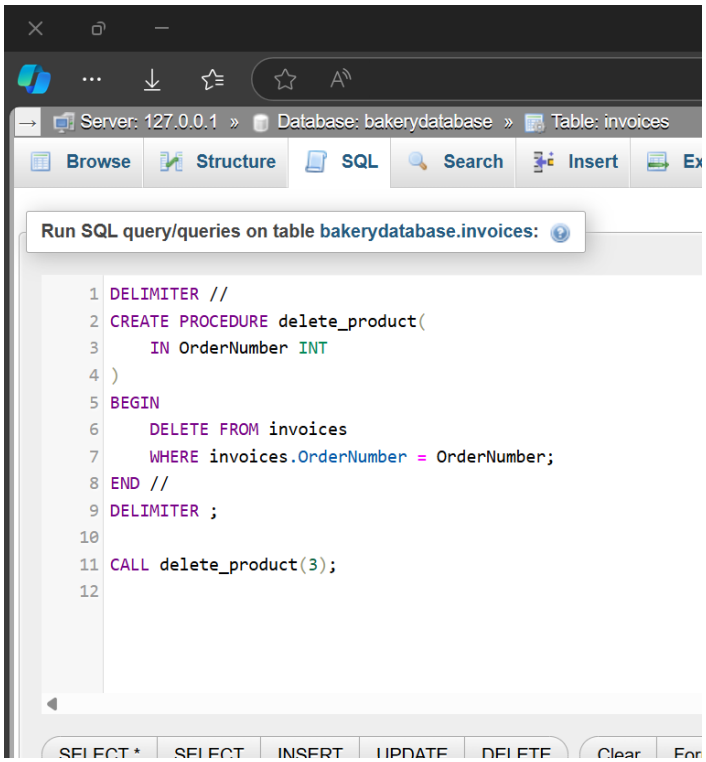
5. Create a Stored Procedure for Updating with calling

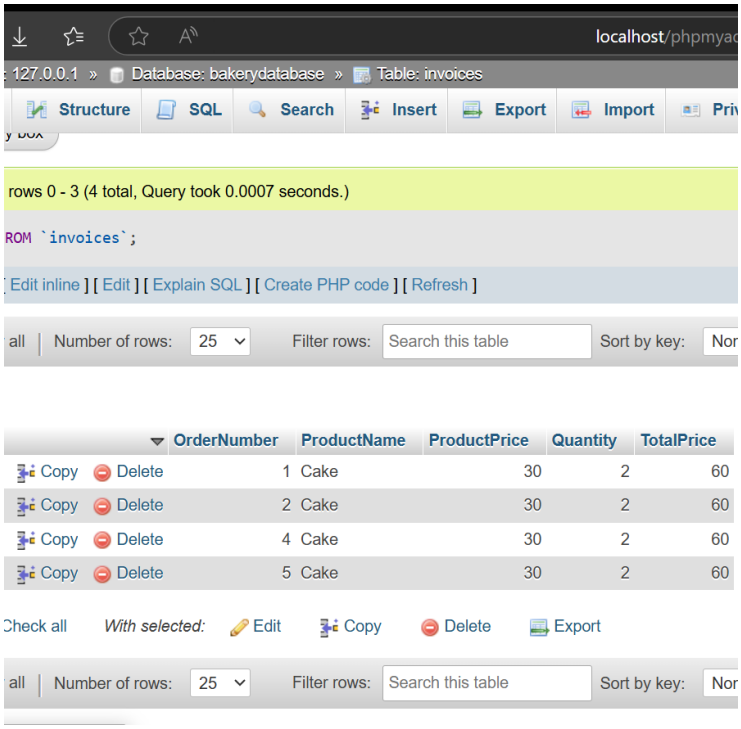


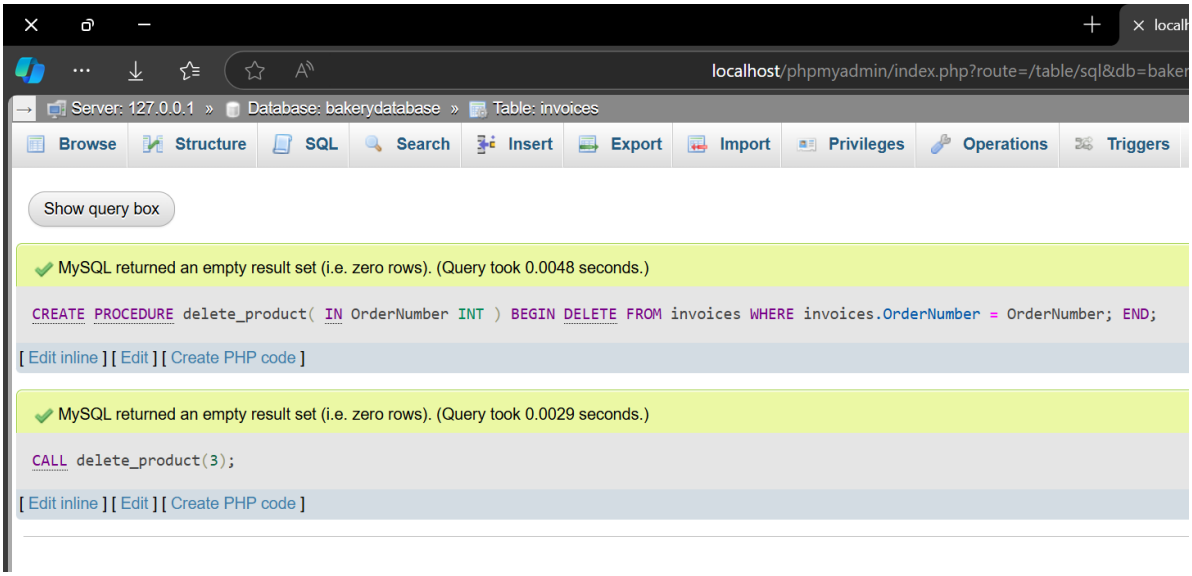




6. Create a Stored Procedure for Deletion with final result

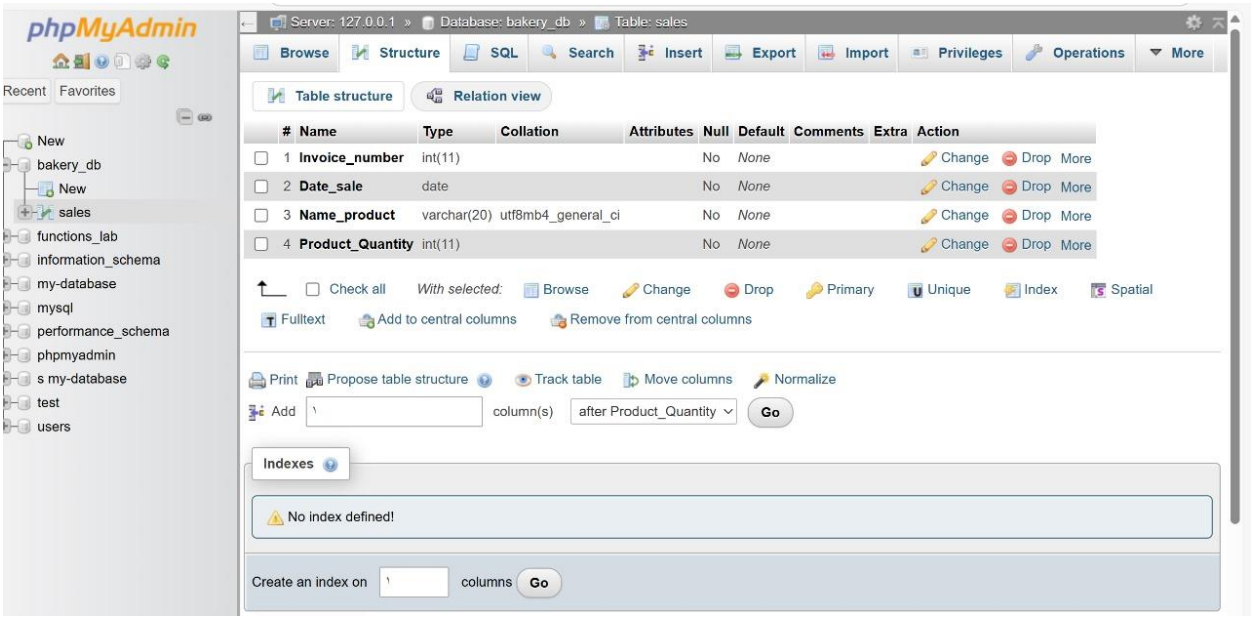




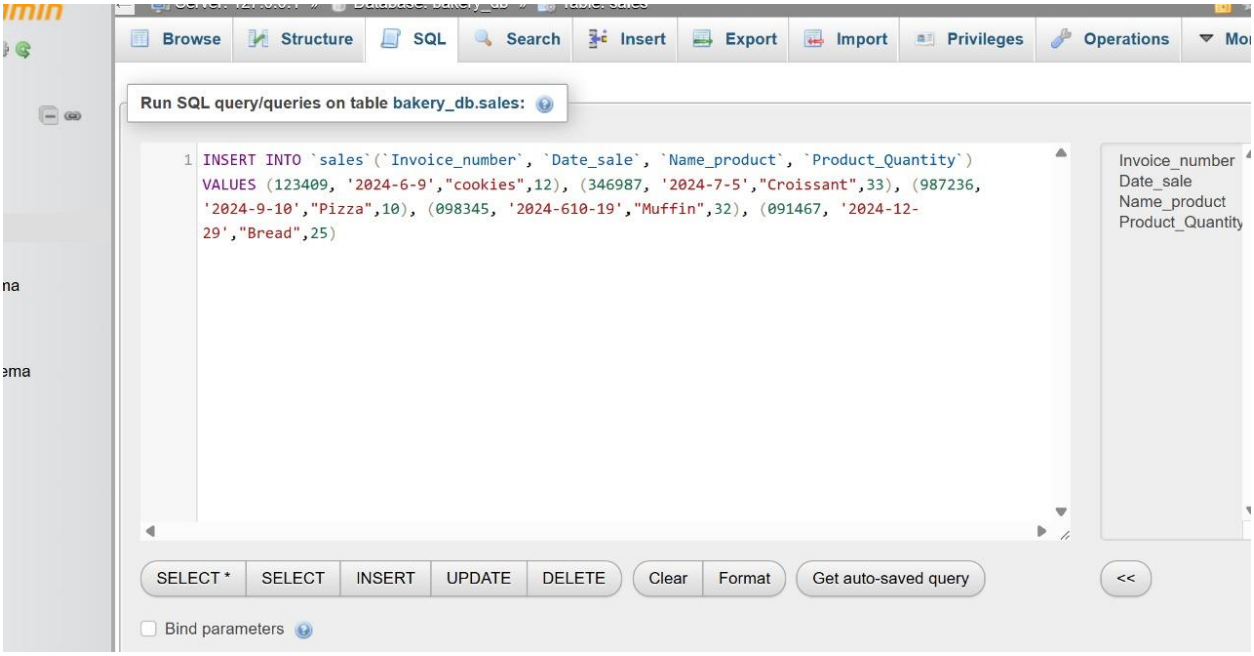


Create and use transaction

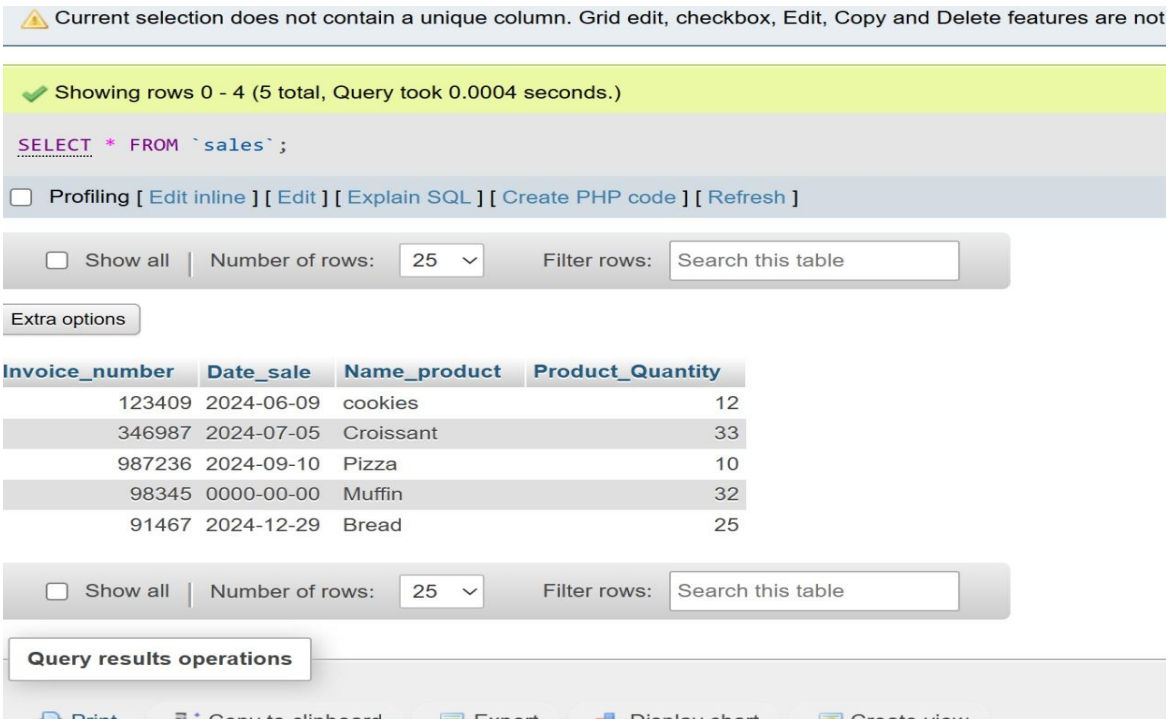
1-Creating Table



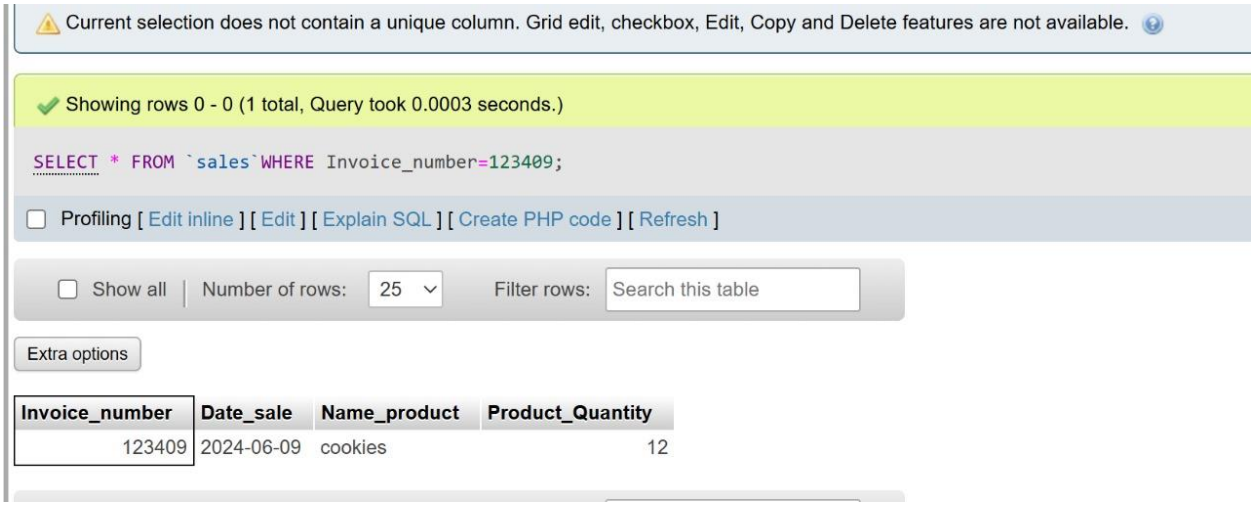
2-Inserting Data



3-Select the table

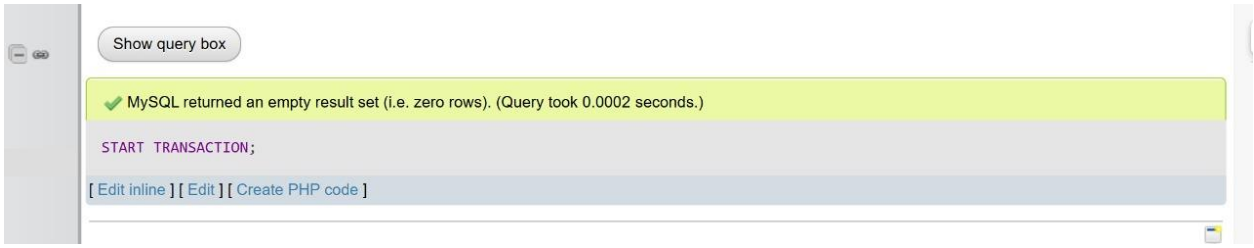


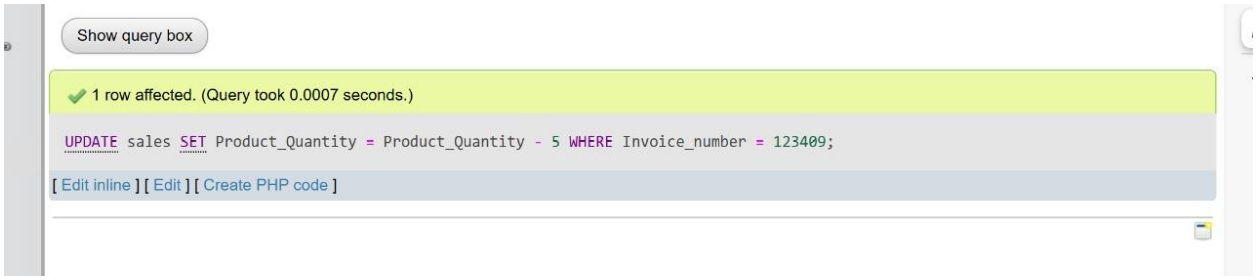
4- Select Where

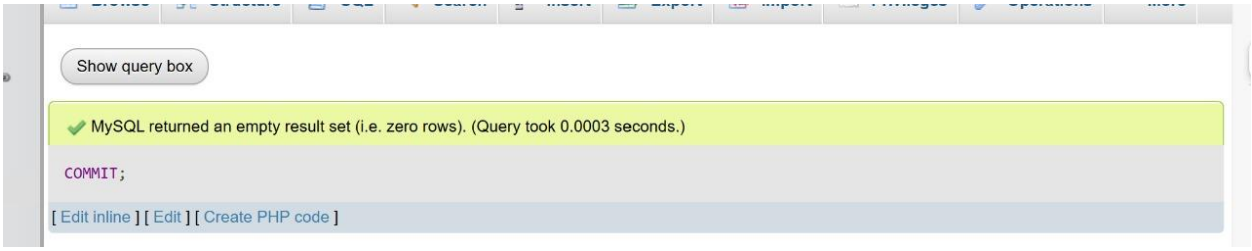


5-Transaction

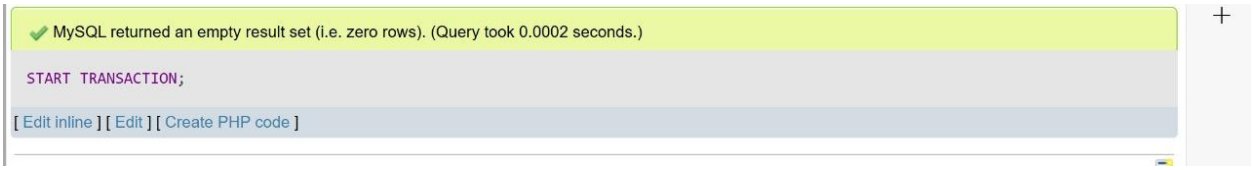
5.1-Simple Transaction

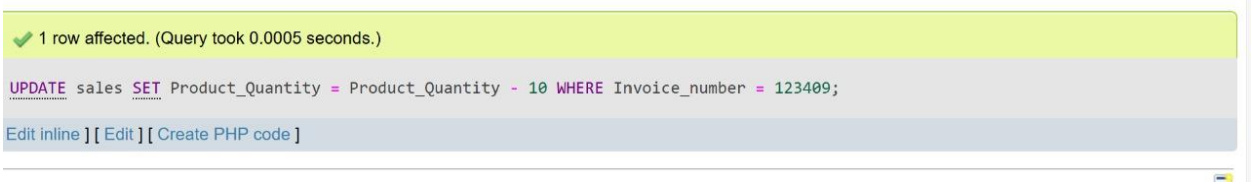


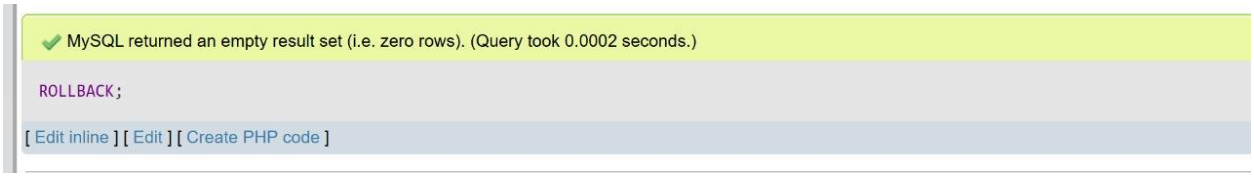




5.2- Rollback Transaction







5.3-Concurrent Updates Without Concurrency Control

5.4-Deadlocks

Create a trigger and test the trigger

1- Create a Database and Table:

Run SQL query/queries on database bakery:

```
CREATE TABLE products (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    product_name VARCHAR(100),
    price DECIMAL(10, 2),
    quantity INT,
    types TEXT
);
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
SELECT * FROM `products`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

product_id	product_name	price	quantity	types

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	product_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	product_name	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	price	decimal(10,2)			Yes	NULL			Change Drop More
4	quantity	int(11)			Yes	NULL			Change Drop More
5	types	text	utf8mb4_general_ci		Yes	NULL			Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext

Print Propose table structure Move columns Normalize

Add 1 column(s) after types Go

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Rename Drop	PRIMARY	BTREE	Yes	No	product_id	5	A	No	

2- Insert Sample Data:

Run SQL query/queries on table bakery.products:

```
INSERT INTO products (product_name, price, quantity, types) VALUES
('Croissant', 11.00,15, 'Classic Corissant, Chocolate Corissant, Cheese Corissant'),
('Cookies', 20.00,40, 'Classic Cookies, Pistachio Cookies, Kinder Cookies'),
('Pizza', 25.00,20, 'Pepperoni Pizza, Margheritte Pizza, Ranch Pizza'),
('Muffin', 15.00,50, 'Blueberry Muffin, Chocolate Muffin, Vanilla Muffin'),
('Beard', 10.00,100, 'White Bread, Bagel, Pita Bread');
```

Showing rows 0 - 4 (5 total, Query took 0.0002 seconds.)

```
SELECT * FROM `products`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	product_id	product_name	price	quantity	types
Edit Copy Delete	1	Croissant	11.00	15	Classic Corissant, Chocolate Corissant, Cheese Cor...
Edit Copy Delete	2	Cookies	20.00	40	Classic Cookies, Pistachio Cookies, Kinder Cookies
Edit Copy Delete	3	Pizza	25.00	20	Pepperoni Pizza, Margheritte Pizza, Ranch Pizza
Edit Copy Delete	4	Muffin	15.00	50	Blueberry Muffin, Chocolate Muffin, Vanilla Muffin
Edit Copy Delete	5	Beard	10.00	100	White Bread, Bagel, Pita Bread

3- Create a Trigger for Price Validation:

Run SQL query/queries on table bakery.products:

```
DELIMITER //

CREATE TRIGGER before_products_update
BEFORE UPDATE ON products
FOR EACH ROW
BEGIN
    IF NEW.price < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Price cannot be negative';
    END IF;
END //

DELIMITER ;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0050 seconds.)

```
CREATE TRIGGER before_products_update BEFORE UPDATE ON products FOR EACH ROW BEGIN IF NEW.price < 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Price cannot be negative'; END IF; END;
```

[Edit inline] [Edit] [Create PHP code]

4- Test the Trigger:

Run SQL query/queries on table bakery.products:

```
1 UPDATE products
2 SET price = -10.00
3 WHERE product_id = 1;
4
```

Error

SQL query: Copy

```
UPDATE products
SET price = -10.00
WHERE product_id = 1;
```

MySQL said:

```
#1644 - Price cannot be negative
```

5- Create a Trigger for Quantity Tracking:

Run SQL query/queries on table bakery.products:

```
1 CREATE TABLE inventory (
2     product_id INT,
3     total_quantity INT,
4     FOREIGN KEY (product_id) REFERENCES products(product_id)
5 );
6
7
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0208 seconds.)

```
CREATE TABLE inventory ( product_id INT, total_quantity INT, FOREIGN KEY (product_id) REFERENCES products(product_id) );
```

[Edit inline] [Edit] [Create PHP code]

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	product_id	int(11)			Yes	NULL			Change Drop More
2	total_quantity	int(11)			Yes	NULL			Change Drop More

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

```
SELECT * FROM `inventory`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

product_id	total_quantity

5.1- insert simple data:

Run SQL query/queries on table bakery.inventory:

```
insert into `inventory` (product_id, total_quantity)  SELECT product_id, quantity FROM products;
```

Showing rows 0 - 4 (5 total, Query took 0.0002 seconds.)

```
SELECT * FROM `inventory`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

product_id	total_quantity
1	15
2	40
3	20
4	50
5	100

6- CREATE TRIGGER after_products_update:

Run SQL query/queries on table bakery.inventory:

```
DELIMITER //
CREATE TRIGGER after_products_update_1
AFTER UPDATE ON products
FOR EACH ROW
BEGIN
    UPDATE inventory
    SET total_quantity = NEW.quantity
    WHERE product_id = OLD.product_id;
END //

DELIMITER ;

```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0056 seconds.)

```
CREATE TRIGGER after_products_update_1 AFTER UPDATE ON products FOR EACH ROW BEGIN UPDATE inventory SET total_quantity = NEW.quantity
WHERE product_id = OLD.product_id; END;
```

[Edit inline] [Edit] [Create PHP code]

6.1- Test the Trigger:

Run SQL query/queries on table bakery.inventory:

```
UPDATE products
SET quantity = 50
WHERE product_id = 1;

```

1 row affected. (Query took 0.0023 seconds.)

```
UPDATE products SET quantity = 50 WHERE product_id = 1;
```

[Edit inline] [Edit] [Create PHP code]

Showing rows 0 - 4 (5 total, Query took 0.0001 seconds.)

```
SELECT * FROM `inventory`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

product_id	total_quantity
1	50
2	40
3	20
4	50
5	100

7- Create a Trigger for Auditing:

Run SQL query/queries on table bakery.inventory:

```
CREATE TABLE product_audit (
    audit_id INT PRIMARY KEY AUTO_INCREMENT,
    product_id INT,
    old_price DECIMAL(10, 2),
    new_price DECIMAL(10, 2),
    old_quantity INT,
    new_quantity INT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

```
SELECT * FROM `product_audit`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

audit_id	product_id	old_price	new_price	old_quantity	new_quantity	timestamp

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	audit_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	product_id	int(11)			Yes	NULL			Change Drop More
3	old_price	decimal(10,2)			Yes	NULL			Change Drop More
4	new_price	decimal(10,2)			Yes	NULL			Change Drop More
5	old_quantity	int(11)			Yes	NULL			Change Drop More
6	new_quantity	int(11)			Yes	NULL			Change Drop More
7	timestamp	timestamp			No	current_timestamp()			Change Drop More

Run SQL query/queries on table bakery.product_audit:

```
DELIMITER //

CREATE TRIGGER after_products_update_2
AFTER UPDATE ON products
FOR EACH ROW
BEGIN
    INSERT INTO product_audit (product_id, old_price, new_price, old_quantity, new_quantity)
    VALUES (OLD.product_id, OLD.price, NEW.price, OLD.quantity, NEW.quantity);
END //

DELIMITER ;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0055 seconds.)

```
CREATE TRIGGER after_products_update_2 AFTER UPDATE ON products FOR EACH ROW BEGIN INSERT INTO product_audit (product_id, old_price, new_price, old_quantity, new_quantity) VALUES (OLD.product_id, OLD.price, NEW.price, OLD.quantity, NEW.quantity); END;
```

[Edit inline] [Edit] [Create PHP code]

7.1- Test the Trigger:

Run SQL query/queries on table bakery.product_audit:

```
UPDATE products
SET price = 1000.00, quantity = 100
WHERE product_id = 1;
```

1 row affected. (Query took 0.0014 seconds.)

```
UPDATE products SET price = 1000.00, quantity = 100 WHERE product_id = 1;
```

[Edit inline] [Edit] [Create PHP code]

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

```
SELECT * FROM `product_audit`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

audit_id	product_id	old_price	new_price	old_quantity	new_quantity	timestamp
1	1	11.00	1000.00	50	100	2024-10-14 23:35:23

Edit Copy Delete

Check all With selected: Edit Copy Delete Export

9- Select the table and select where:

Showing rows 0 - 4 (5 total, Query took 0.0001 seconds.)

```
SELECT * FROM `products`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	product_id	product_name	price	quantity	types
Edit Copy Delete	1	Croissant	1000.00	100	Classic Corissant, Chocolate Corissant, Cheese Cor...
Edit Copy Delete	2	Cookies	20.00	40	Classic Cookies, Pistachio Cookies, Kinder Cookies
Edit Copy Delete	3	Pizza	25.00	20	Pepperoni Pizza, Margheritte Pizza, Ranch Pizza
Edit Copy Delete	4	Muffin	15.00	50	Blueberry Muffin, Chocolate Muffin, Vanilla Muffin
Edit Copy Delete	5	Beard	10.00	100	White Bread, Bagel, Pita Bread

Showing rows 0 - 3 (4 total, Query took 0.0002 seconds.)

```
SELECT * FROM `products` WHERE price < 50;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	product_id	product_name	price	quantity	types
Edit Copy Delete	2	Cookies	20.00	40	Classic Cookies, Pistachio Cookies, Kinder Cookies
Edit Copy Delete	3	Pizza	25.00	20	Pepperoni Pizza, Margheritte Pizza, Ranch Pizza
Edit Copy Delete	4	Muffin	15.00	50	Blueberry Muffin, Chocolate Muffin, Vanilla Muffin
Edit Copy Delete	5	Beard	10.00	100	White Bread, Bagel, Pita Bread

Check all *With selected:* Edit Copy Delete Export