

Umm AL-Qura University

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	Create Order Invoice Table, Insert Data to this Table, Create Procedure and call it.
Wejdan Attia AlZahrani	444004760	5	Create Sales Table, Insert Data to this Table, Create and use transaction.
Halimah Abutaleb Alsayed	444005535	3	Create 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

The screenshot shows two windows from MySQL Workbench.

The top window is titled "Run SQL query/queries on database bakery_db:" and contains the following SQL code:

```
1 CREATE TABLE employee (
2     employee_id INT PRIMARY KEY AUTO_INCREMENT,
3     first_name VARCHAR(50),
4     last_name VARCHAR(50),
5     position VARCHAR(50),
6     hire_date DATE,
7     salary DECIMAL(10, 2)
8 );
```

The bottom window is titled "Table: employee" and shows the results of a SELECT query:

```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0009 seconds.)
```

```
SELECT * FROM `employee`
```

Below the query results, the table structure is listed:

employee_id	first_name	last_name	position	hire_date	salary
-------------	------------	-----------	----------	-----------	--------

Under the table structure, there are "Query results operations" and a "Create view" button.

2. Inserting Data

2.2 Insert Data into Employee Table

The screenshot shows the MySQL Workbench interface for the 'bakery_db' database. The 'employee' table is selected. In the SQL tab, an INSERT query is run:

```
1 INSERT INTO employee (first_name, last_name, position, hire_date, salary)
2 VALUES
3 ('Fahad', 'Al-Qahtani', 'Baker', '2023-01-15', 5000.00),
4 ('Sara', 'Al-Harbi', 'Cashier', '2022-11-20', 3500.00),
5 ('Khalid', 'Al-Saud', 'Manager', '2021-06-10', 10000.00),
6 ('Aisha', 'Al-Shammari', 'Assistant', '2023-02-25', 3000.00),
7 ('Noura', 'Al-Otaibi', 'Salesperson', '2023-03-05', 4000.00);
8
```

The results pane shows the inserted data:

employee_id	first_name	last_name	position	hire_date	salary
1	Fahad	Al-Qahtani	Baker	2023-01-15	5000.00
2	Sara	Al-Harbi	Cashier	2022-11-20	3500.00
3	Khalid	Al-Saud	Manager	2021-06-10	10000.00
4	Aisha	Al-Shammari	Assistant	2023-02-25	3000.00
5	Noura	Al-Otaibi	Salesperson	2023-03-05	4000.00

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.

The screenshot shows the phpMyAdmin interface for the 'employee' table. The results of the query `SELECT first_name, last_name, hire_date FROM employee WHERE hire_date > '2023-01-01';` are displayed. The results are:

	first_name	last_name	hire_date
<input type="checkbox"/>	Fahad	Al-Qahtani	2023-01-15
<input type="checkbox"/>	Aisha	Al-Shammari	2023-02-25
<input type="checkbox"/>	Noura	Al-Otaibi	2023-03-05

Below the table, there are buttons for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'.

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).

The screenshot shows the phpMyAdmin interface for the 'employee' table. The results of the query `SELECT first_name, last_name, salary FROM employee ORDER BY salary DESC;` are displayed. The results are:

	first_name	last_name	salary
<input type="checkbox"/>	Khalid	Al-Saud	10000.00
<input type="checkbox"/>	Fahad	Al-Qahtani	5000.00
<input type="checkbox"/>	Noura	Al-Otaibi	4000.00
<input type="checkbox"/>	Sara	Al-Harbi	3500.00
<input type="checkbox"/>	Aisha	Al-Shammari	3000.00

Below the table, there are buttons for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'.

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".

Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)

```
SELECT first_name, last_name, salary FROM employee WHERE last_name = 'Al-Qahtani';
```

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

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

Extra options

	first_name	last_name	salary
<input type="checkbox"/>	Fahad	Al-Qahtani	5000.00

Check all With selected: Edit Copy Delete Export

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

Query results operations

4. Create an index and function

4.1. Create Index

index on the employee's last_name :

Server: 127.0.0.1 » Database: bakery_db » Table: employee

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Run SQL query/queries on table bakery_db.employee:

```
1 CREATE INDEX idx_last_name ON employee (last_name);
2
```

employee_id
first_name
last_name
position
hire_date
salary

phpMyAdmin

Recent Favorites

New

bakery_db

New

employee

Columns

Indexes

New

idx_last_name

PRIMARY

Show query box

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0214 seconds.)

```
CREATE INDEX idx_last_name ON employee (last_name);
```

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

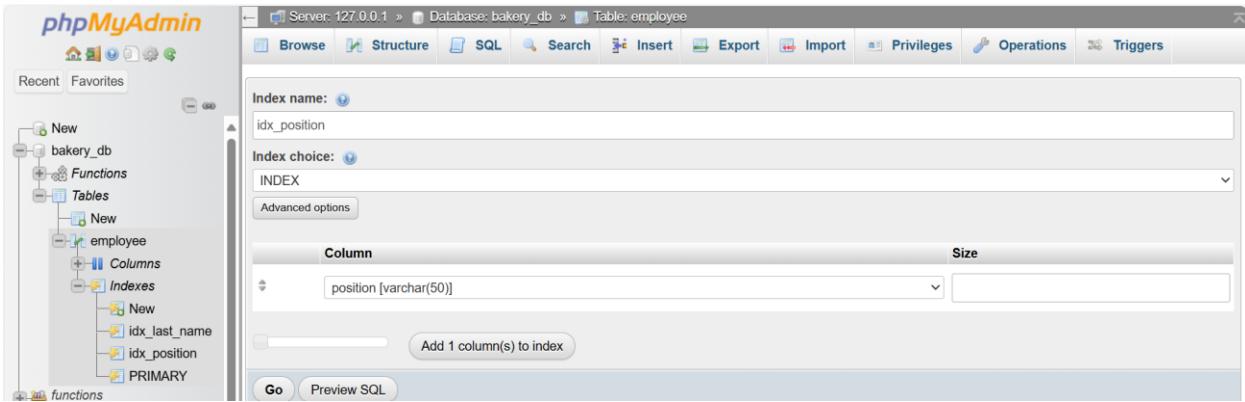
The screenshot shows the phpMyAdmin interface for the 'bakery_db' database. In the left sidebar, under the 'Tables' section, the 'employee' table is selected. In the main panel, the 'Indexes' tab is active. A new index is being created with the name 'idx_last_name'. The 'Index choice' dropdown is set to 'INDEX'. Under the 'Column' section, 'last_name [varchar(50)]' is listed. There is also a button to 'Add 1 column(s) to index'.

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

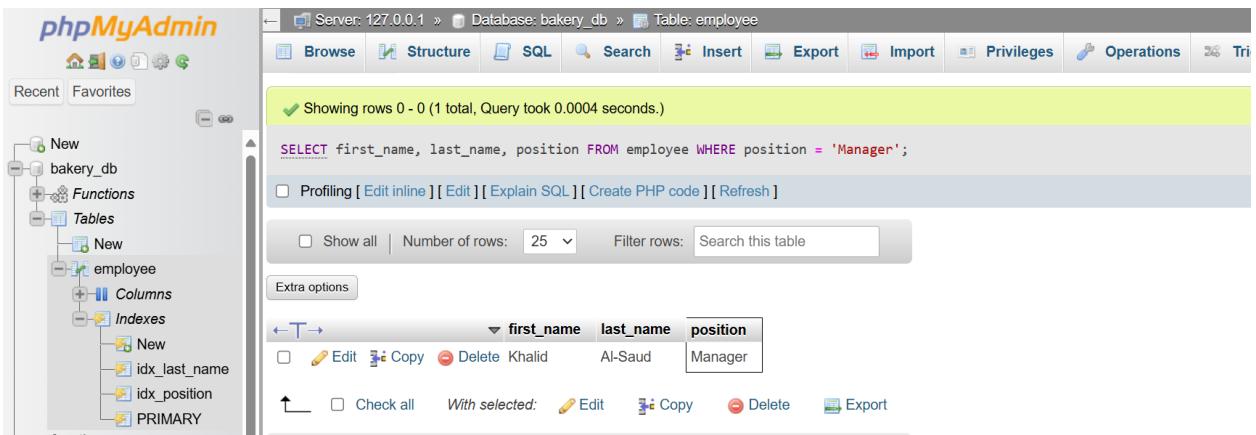
The screenshot shows the phpMyAdmin interface for the 'bakery_db' database. The 'employee' table is selected. In the main panel, the 'Browse' tab is active. A query is run: 'SELECT first_name, last_name, salary FROM employee WHERE last_name = 'Al-Qahtani''. The results show one row: Fahad Al-Qahtani with a salary of 5000.00.

Create an Index on employee position

The screenshot shows the phpMyAdmin interface for the 'bakery_db' database. The 'employee' table is selected. In the main panel, the 'Indexes' tab is active. A new index is being created with the name 'idx_position'. The 'Index choice' dropdown is set to 'INDEX'. Under the 'Column' section, 'position' is listed. There is also a button to 'Add 1 column(s) to index'.



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.

phpMyAdmin

Server: 127.0.0.1 » Database: bakery_db » Table: employee

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Run SQL query/queries on table bakery_db.employee:

```

1 DELIMITER //
2
3 CREATE FUNCTION calculate_bonus(emp_id INT)
4 RETURNS DECIMAL(10, 2)
5 DETERMINISTIC
6 BEGIN
7     DECLARE yearly_bonus DECIMAL(10, 2);
8
9     SELECT salary * 0.10 INTO yearly_bonus
10    FROM employee
11   WHERE employee_id = emp_id;
12
13     RETURN yearly_bonus;
14 END //
15
16 DELIMITER ;
17

```

employee_id
first_name
last_name
position
hire_date
salary

Server: 127.0.0.1 » Database: bakery_db » Table: employee

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

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

```

CREATE FUNCTION calculate_bonus(emp_id INT) RETURNS DECIMAL(10, 2) DETERMINISTIC BEGIN DECLARE yearly_bonus DECIMAL(10, 2); SELECT salary
* 0.10 INTO yearly_bonus FROM employee WHERE employee_id = emp_id; RETURN yearly_bonus; END;

```

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

phpMyAdmin

Server: 127.0.0.1 » Database: bakery_db

Structure SQL Search Query Export Import Operations Privileges Routines More

Routine name: calculate_bonus

Type: FUNCTION

Name	Type	Length/Values	Options
emp_id	INT		Charset

Add parameter Remove last parameter

Return type: DECIMAL

Return length/values: 10,2

Return options: Charset

BEGIN

```

DECLARE yearly_bonus DECIMAL(10, 2);

SELECT salary * 0.10 INTO

```

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.

phpMyAdmin

Server: 127.0.0.1 » Database: bakery_db » Table: employee

Browse Structure SQL Search Insert Export Import Privileges Operations

Run SQL query/queries on table bakery_db.employee:

```
1 SELECT calculate_bonus(1) AS bonus;
```

emp first_name last_name position hire_date salary

phpMyAdmin

Server: 127.0.0.1 » Database: bakery_db » Table: employee

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0011 seconds.)

```
SELECT calculate_bonus(1) AS bonus;
```

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

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

Extra options

bonus
500.00

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

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

`SELECT first_name, last_name, salary, calculate_bonus(employee_id) AS bonus FROM employee;`

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

	first_name	last_name	salary	bonus
<input type="checkbox"/>	Edit	Copy	Delete	Fahad Al-Qahtani 5000.00 500.00
<input type="checkbox"/>	Edit	Copy	Delete	Sara Al-Harbi 3500.00 350.00
<input type="checkbox"/>	Edit	Copy	Delete	Khalid Al-Saud 10000.00 1000.00
<input type="checkbox"/>	Edit	Copy	Delete	Aisha Al-Shammari 3000.00 300.00
<input type="checkbox"/>	Edit	Copy	Delete	Noura Al-Otaibi 4000.00 400.00

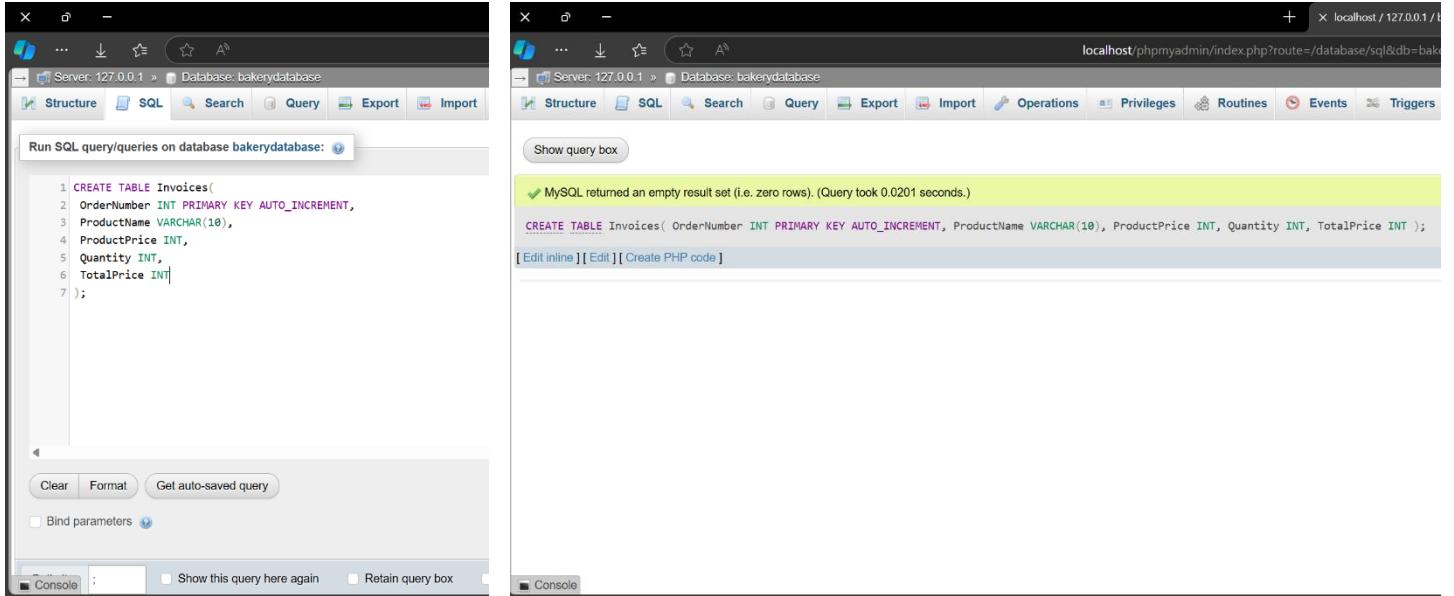
Check all With selected: Edit Copy Delete Export

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

Create a Procedure and Call it

1. Create table

1.1. Create Order Invoice table



The image shows two side-by-side phpMyAdmin interface windows. Both windows have the URL `localhost/phpmyadmin/index.php?route=/database/sql&db=bakerydatabase` and the server is set to `127.0.0.1`.

The left window's SQL query box contains the following code:

```
1 CREATE TABLE Invoices(
2     OrderNumber INT PRIMARY KEY AUTO_INCREMENT,
3     ProductName VARCHAR(10),
4     ProductPrice INT,
5     Quantity INT,
6     TotalPrice INT
7 );
```

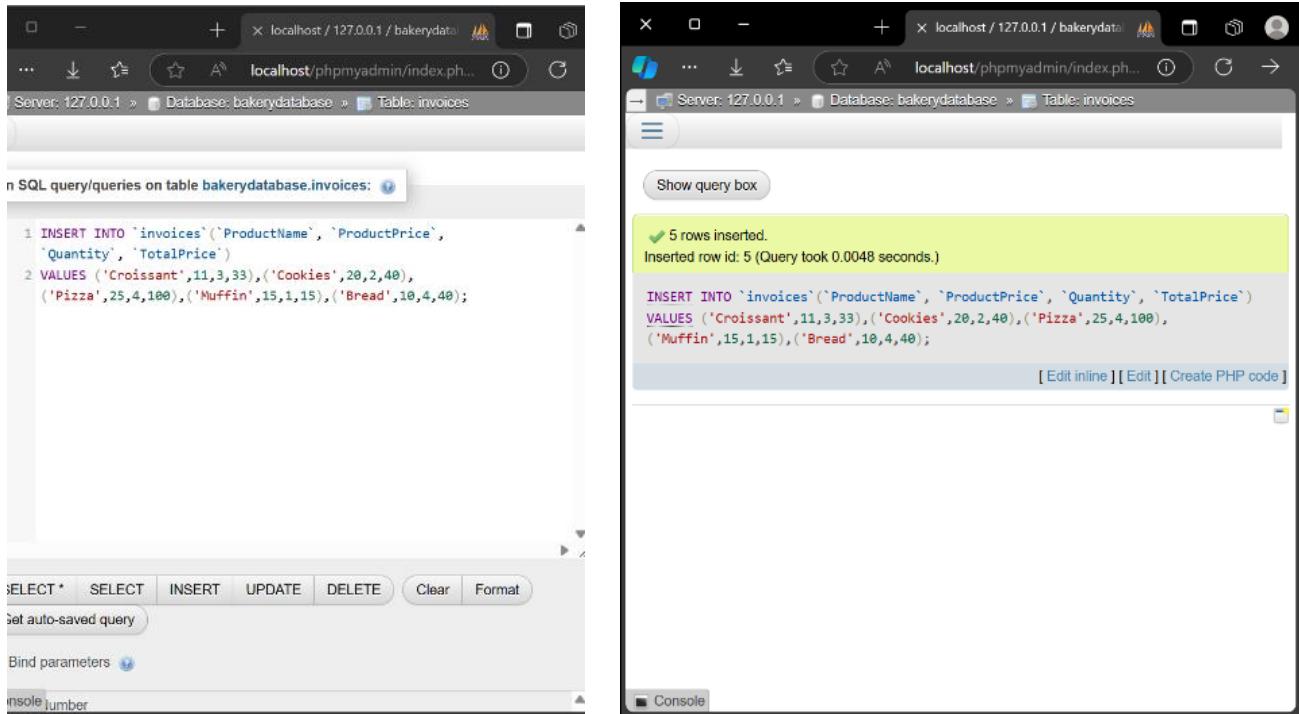
The right window's SQL query box contains the same code, followed by a success message:

```
CREATE TABLE Invoices( OrderNumber INT PRIMARY KEY AUTO_INCREMENT, ProductName VARCHAR(10), ProductPrice INT, Quantity INT, TotalPrice INT );
```

Below the message, there are links for [Edit inline], [Edit], and [Create PHP code].

2. Insert data into table

2.1. Insert data into Order Invoice table



The image shows two side-by-side phpMyAdmin interface windows. Both windows have the URL `localhost/phpmyadmin/index.php?route=/database/sql&db=bakerydatabase` and the server is set to `127.0.0.1`. The left window is titled "Table: invoices".

The left window's SQL query box contains the following code:

```
1 INSERT INTO `invoices`(`ProductName`, `ProductPrice`,
2 `Quantity`, `TotalPrice`)
3 VALUES ('Croissant',11,3,33),('Cookies',20,2,40),
4 ('Pizza',25,4,100),('Muffin',15,1,15),('Bread',10,4,40);
```

The right window's SQL query box contains the same code, followed by a success message:

```
INSERT INTO `invoices`(`ProductName`, `ProductPrice`, `Quantity`, `TotalPrice`)
VALUES ('Croissant',11,3,33),('Cookies',20,2,40),('Pizza',25,4,100),
('Muffin',15,1,15),('Bread',10,4,40);
```

Below the message, there are links for [Edit inline], [Edit], and [Create PHP code].

3. Select and Select where

The image shows two side-by-side screenshots of the phpMyAdmin interface. Both are connected to the 'bakedatabase' database and the 'invoices' table.

Screenshot 1 (Left): The query executed is:

```
SELECT * FROM `invoices` WHERE TotalPrice=40;
```

The results show two rows:

OrderNumber	ProductName	ProductPrice	Quantity	TotalPrice
2	Cookies	20	2	40
5	Bread	10	4	40

Screenshot 2 (Right): The query executed is:

```
SELECT * FROM `invoices`;
```

The results show five rows:

OrderNumber	ProductName	ProductPrice	Quantity	TotalPrice
1	Croissant	11	3	33
2	Cookies	20	2	40
3	Pizza	25	4	100
4	Muffin	15	1	15
5	Bread	10	4	40

4. Create a Stored Procedure for Insertion

The image shows a screenshot of the phpMyAdmin interface. A SQL query is being run on the 'invoices' table of the 'bakedatabase' database.

The query is:

```
1 DELIMITER //
2 CREATE PROCEDURE insert_products(
3     IN ProductName VARCHAR(10),
4     IN ProductPrice INT,
5     IN Quantity INT,
6     IN TotalPrice INT
7 )
8 BEGIN
9     INSERT INTO invoices (ProductName, ProductPrice, Quantity,
10                         TotalPrice)
11     VALUES (ProductName, ProductPrice, Quantity, TotalPrice);
12 END //
13 DELIMITER ;
```

The interface includes a toolbar with buttons for SELECT, INSERT, UPDATE, DELETE, and a status bar at the bottom.

5. Create a Stored Procedure for Updating with calling

The image shows two adjacent browser windows side-by-side. Both windows are titled "localhost / 127.0.0.1 / bakerydata" and "Database: bakerydatabase" with "Table: invoices".

The left window displays the SQL code for creating a stored procedure:

```
1 DELIMITER //
2 CREATE PROCEDURE update_product(
3     IN OrderNumber INT,
4     IN ProductName VARCHAR(10),
5     IN ProductPrice INT,
6     IN Quantity INT,
7     IN TotalPrice INT)
8 BEGIN
9     UPDATE invoices
10    SET ProductName = ProductName,
11        ProductPrice = ProductPrice,
12        Quantity = Quantity,
13        TotalPrice = TotalPrice
14   WHERE OrderNumber = OrderNumber;
15 END //
16 DELIMITER ;
17 CALL update_product(3, 'Cake', 30, 2, 60);
18
```

The right window shows the results of running the stored procedure:

```
CREATE PROCEDURE update_product( IN OrderNumber INT, IN ProductName VARCHAR(10), IN ProductPrice INT, IN Quantity INT, IN TotalPrice INT) BEGIN UPDATE invoices SET ProductName = ProductName, ProductPrice = ProductPrice, Quantity = Quantity, TotalPrice = TotalPrice WHERE OrderNumber = OrderNumber; END;
```

Below this, another query is run:

```
CALL update_product(3, 'Cake', 30, 2, 60);
```

This image shows a single browser window with the URL "localhost/phpmyadmin/index.php?route=/table/sql&db=bakerydatabase&table=invoices". The window title is "localhost / 127.0.0.1 / bakerydata" and "Database: bakerydatabase" with "Table: invoices".

The top navigation bar includes "Browse", "Structure", "SQL", "Search", "Insert", "Export", "Import", "Privileges", "Operations", and "Triggers".

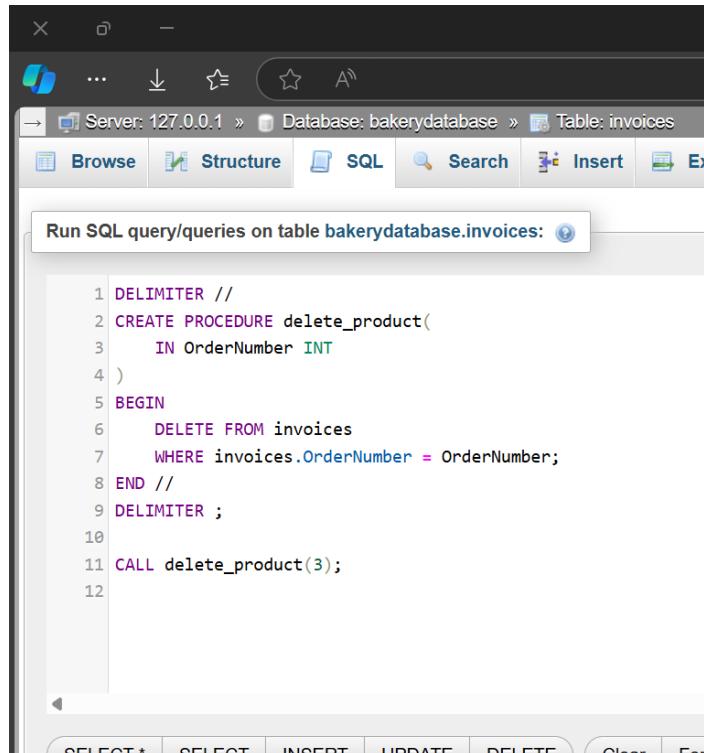
The main area shows the results of a SELECT query:

```
SELECT * FROM `invoices`;
```

Below the results, there is a table titled "Showing rows 0 - 4 (total, Query took 0.0003 seconds.)". The table contains the following data:

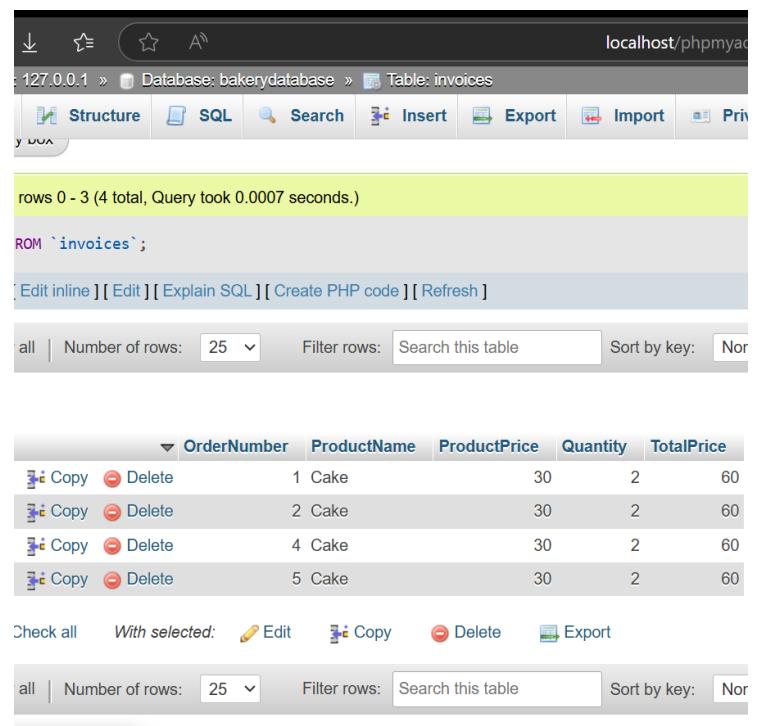
	OrderNumber	ProductName	ProductPrice	Quantity	TotalPrice
<input type="checkbox"/>	1	Cake	30	2	60
<input type="checkbox"/>	2	Cake	30	2	60
<input type="checkbox"/>	3	Cake	30	2	60
<input type="checkbox"/>	4	Cake	30	2	60
<input type="checkbox"/>	5	Cake	30	2	60

6. Create a Stored Procedure for Deletion with final result



The screenshot shows the MySQL Workbench interface. In the left pane, under 'Server: 127.0.0.1 > Database: bakerydatabase > Table: invoices', there is a SQL editor window. The SQL code is:

```
1 DELIMITER //
2 CREATE PROCEDURE delete_product(
3     IN OrderNumber INT
4 )
5 BEGIN
6     DELETE FROM invoices
7     WHERE invoices.OrderNumber = OrderNumber;
8 END //
9 DELIMITER ;
10
11 CALL delete_product(3);
12
```



The screenshot shows the phpMyAdmin interface for the 'invoices' table. The results of the query execution are displayed in a green box:

rows 0 - 3 (4 total, Query took 0.0007 seconds.)

Query results:

```
ROM `invoices`;
```

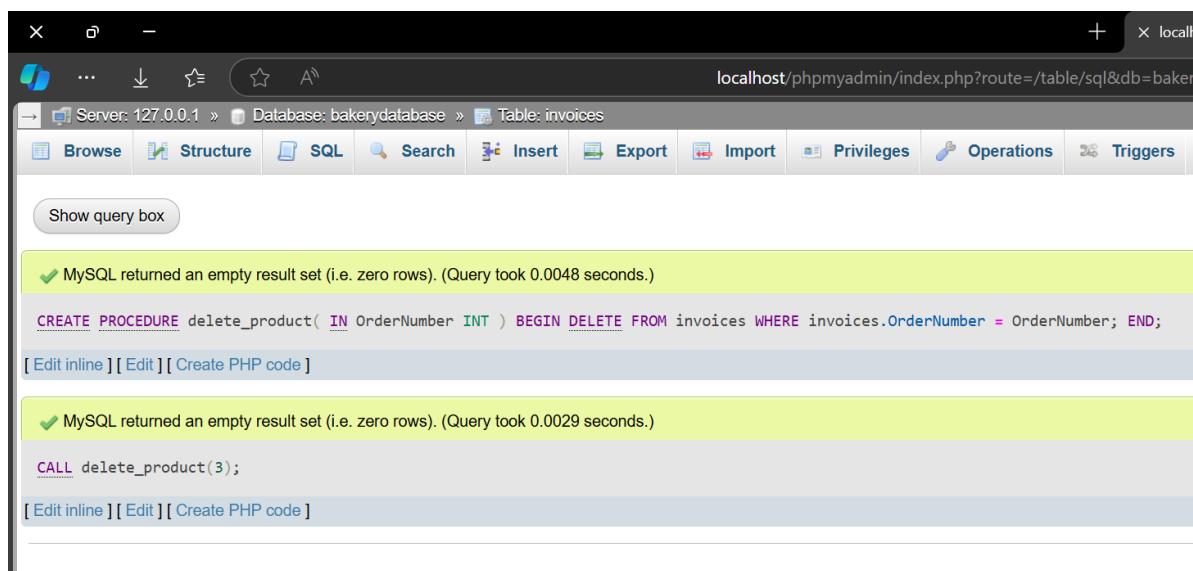
[Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

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

	OrderNumber	ProductName	ProductPrice	Quantity	TotalPrice
Copy	1	Cake	30	2	60
Copy	2	Cake	30	2	60
Copy	4	Cake	30	2	60
Copy	5	Cake	30	2	60

Check all With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

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



The screenshot shows the phpMyAdmin interface for the 'invoices' table. The results of the query execution are displayed in a green box:

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0048 seconds.)

CREATE PROCEDURE delete_product(IN OrderNumber INT) BEGIN DELETE FROM invoices WHERE invoices.OrderNumber = OrderNumber; END;

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

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0029 seconds.)

CALL delete_product(3);

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

Create and use transaction

1-Creating Table

The screenshot shows the phpMyAdmin interface for the 'sales' table in the 'bakery_db' database. The table structure is defined as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Invoice_number	int(11)			No	None			Change Drop More
2	Date_sale	date			No	None			Change Drop More
3	Name_product	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
4	Product_Quantity	int(11)			No	None			Change Drop More

Below the table structure, there are buttons for 'Check all', 'With selected:', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', 'Fulltext', 'Add to central columns', and 'Remove from central columns'. There are also links for 'Print', 'Propose table structure', 'Track table', 'Move columns', and 'Normalize'. A search bar for 'Add' and a dropdown for 'column(s)' and 'after Product_Quantity' are present. At the bottom, there is a 'Indexes' section with a note 'No index defined!' and a 'Create an index on' button.

2-Inserting Data

The screenshot shows the phpMyAdmin interface running an SQL query on the 'sales' table. The query is:

```
1 INSERT INTO `sales`(`Invoice_number`, `Date_sale`, `Name_product`, `Product_Quantity`)
VALUES (123409, '2024-6-9', "cookies", 12), (346987, '2024-7-5', "Croissant", 33), (987236,
'2024-9-10', "Pizza", 10), (098345, '2024-610-19', "Muffin", 32), (091467, '2024-12-
29', "Bread", 25)
```

To the right of the query, a preview of the inserted data is shown in a grid:

Invoice_number	Date_sale	Name_product	Product_Quantity
123409	2024-6-9	cookies	12
346987	2024-7-5	Croissant	33
987236	2024-9-10	Pizza	10
098345	2024-610-19	Muffin	32
091467	2024-12-29	Bread	25

At the bottom of the interface, there are buttons for 'SELECT *', 'SELECT', 'INSERT', 'UPDATE', 'DELETE', 'Clear', 'Format', 'Get auto-saved query', and a checkbox for 'Bind parameters'.

3-Select the table

⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)

```
SELECT * FROM `sales`;
```

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

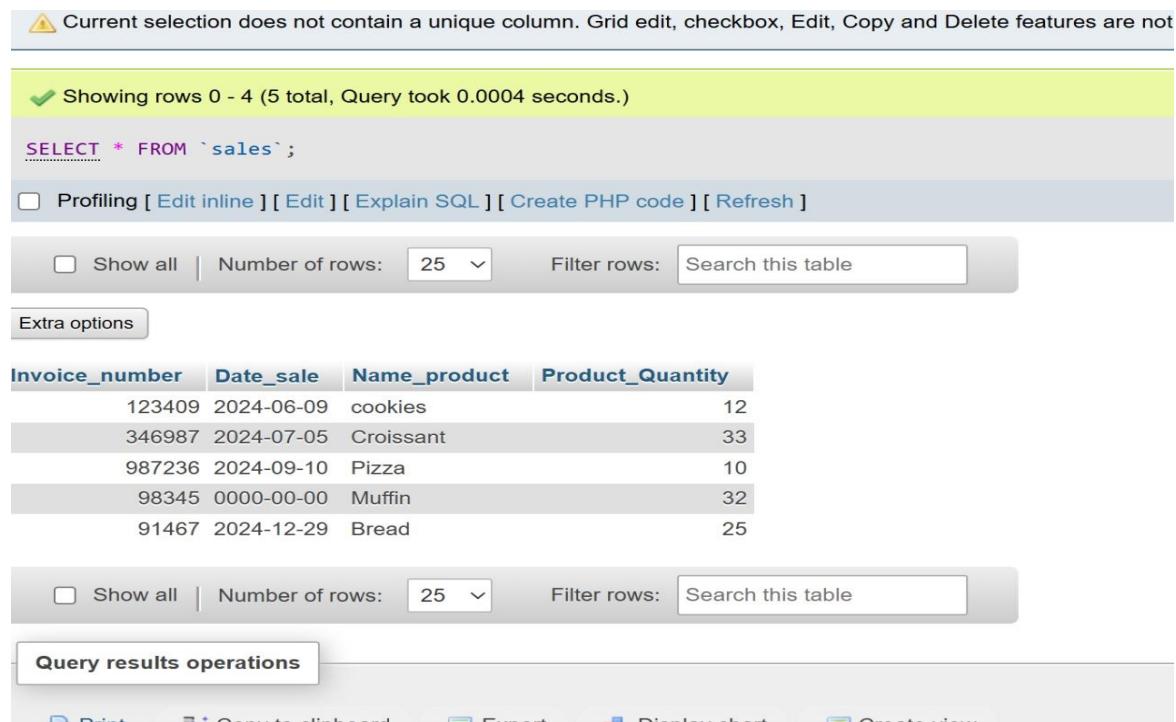
Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Invoice_number	Date_sale	Name_product	Product_Quantity
123409	2024-06-09	cookies	12
346987	2024-07-05	Croissant	33
987236	2024-09-10	Pizza	10
98345	0000-00-00	Muffin	32
91467	2024-12-29	Bread	25

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

Query results operations



4- Select Where

⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

```
SELECT * FROM `sales` WHERE Invoice_number=123409;
```

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

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

Extra options

Invoice_number	Date_sale	Name_product	Product_Quantity
123409	2024-06-09	cookies	12

5-Transaction

5.1-Simple Transaction

The screenshot shows three separate query results in a MySQL Workbench interface. Each result is preceded by a green checkmark icon and a message indicating the number of rows affected or a note about an empty result set. The queries themselves are shown in a grey input area below each result.

- Result 1: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)
Query: START TRANSACTION;
- Result 2: 1 row affected. (Query took 0.0007 seconds.)
Query: UPDATE sales SET Product_Quantity = Product_Quantity - 5 WHERE Invoice_number = 123409;
- Result 3: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
Query: COMMIT;

5.2- Rollback Transaction

The screenshot shows three separate query results in a MySQL Workbench interface. Each result is preceded by a green checkmark icon and a message indicating the number of rows affected or a note about an empty result set. The queries themselves are shown in a grey input area below each result.

- Result 1: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)
Query: START TRANSACTION;
- Result 2: 1 row affected. (Query took 0.0005 seconds.)
Query: UPDATE sales SET Product_Quantity = Product_Quantity - 10 WHERE Invoice_number = 123409;
- Result 3: MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)
Query: ROLLBACK;

5.3-Concurrent Updates Without Concurrency Control

The screenshot shows a MySQL query interface with the following sequence of queries and responses:

- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
- START TRANSACTION;
- [Edit inline] [Edit] [Create PHP code]
- 1 row affected. (Query took 0.0006 seconds.)
- UPDATE sales SET Product_Quantity = Product_Quantity - 5 WHERE Invoice_number = 123409;
- [Edit inline] [Edit] [Create PHP code]
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0032 seconds.)
- COMMIT;
- [Edit inline] [Edit] [Create PHP code]

5.4-Deadlocks

The screenshot shows a MySQL query interface with the following sequence of queries and responses:

- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)
- START TRANSACTION;
- [Edit inline] [Edit] [Create PHP code]
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
- LOCK TABLES sales WRITE;
- [Edit inline] [Edit] [Create PHP code]
- 1 row affected. (Query took 0.0007 seconds.)
- UPDATE sales SET Product_Quantity = Product_Quantity + 10 WHERE Invoice_number = 123409;
- [Edit inline] [Edit] [Create PHP code]
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)
- UNLOCK TABLES;
- [Edit inline] [Edit] [Create PHP code]
- MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
- COMMIT;
- [Edit inline] [Edit] [Create PHP code]

Create a trigger and test the trigger

1- Create a Database and Table:

Run SQL query/queries on database **bakery**: [?](#)

```
1 CREATE TABLE products (
2     product_id INT PRIMARY KEY AUTO_INCREMENT,
3     product_name VARCHAR(100),
4     price DECIMAL(10, 2),
5     quantity INT,
6     types TEXT
7 );
8
```

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
<input type="checkbox"/>	1 product_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 product_name	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	3 price	decimal(10,2)			Yes	NULL			Change Drop More
<input type="checkbox"/>	4 quantity	int(11)			Yes	NULL			Change Drop More
<input type="checkbox"/>	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 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**:

```
1 INSERT INTO products (product_name, price, quantity, types) VALUES
2 ('Croissant', 11.00, 15, 'Classic Corissant, Chocolate Corissant, Cheese Corissant'),
3 ('Cookies', 20.00, 40, 'Classic Cookies, Pistachio Cookies, Kinder Cookies'),
4 ('Pizza', 25.00, 20, 'Pepperoni Pizza, Margheritte Pizza, Ranch Pizza'),
5 ('Muffin', 15.00, 50, 'Blueberry Muffin, Chocolate Muffin, Vanilla Muffin'),
6 ('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
<input type="checkbox"/>	1	Croissant	11.00	15	Classic Corissant, Chocolate Corissant, Cheese Cor...
<input type="checkbox"/>	2	Cookies	20.00	40	Classic Cookies, Pistachio Cookies, Kinder Cookies
<input type="checkbox"/>	3	Pizza	25.00	20	Pepperoni Pizza, Margheritte Pizza, Ranch Pizza
<input type="checkbox"/>	4	Muffin	15.00	50	Blueberry Muffin, Chocolate Muffin, Vanilla Muffin
<input type="checkbox"/>	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**:

```
1 DELIMITER //
2
3 CREATE TRIGGER before_products_update
4 BEFORE UPDATE ON products
5 FOR EACH ROW
6 BEGIN
7     IF NEW.price < 0 THEN
8         SIGNAL SQLSTATE '45000'
9         SET MESSAGE_TEXT = 'Price cannot be negative';
10    END IF;
11 END //
```

13 DELIMITER ;
14 |

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
<input type="checkbox"/>	1 product_id	int(11)			Yes	NULL		Change Drop More	
<input type="checkbox"/>	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`:

```
1 insert into `inventory` (product_id, total_quantity) SELECT product_id, quantity FROM products;
2
3
```

✓ 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: 

```
1 DELIMITER //
2 CREATE TRIGGER after_products_update_1
3 AFTER UPDATE ON products
4 FOR EACH ROW
5 BEGIN
6     UPDATE inventory
7         SET total_quantity = NEW.quantity
8     WHERE product_id = OLD.product_id;
9 END //
10
11 DELIMITER ;
12 |
```

 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: 

```
1 UPDATE products
2 SET quantity = 50
3 WHERE product_id = 1;
4
```

 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**:

```
1 CREATE TABLE product_audit (
2     audit_id INT PRIMARY KEY AUTO_INCREMENT,
3     product_id INT,
4     old_price DECIMAL(10, 2),
5     new_price DECIMAL(10, 2),
6     old_quantity INT,
7     new_quantity INT,
8     timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
9 );
```

10

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
<input type="checkbox"/>	1 audit_id	int(11)		No	None		AUTO_INCREMENT		Change Drop More
<input type="checkbox"/>	2 product_id	int(11)		Yes	NULL				Change Drop More
<input type="checkbox"/>	3 old_price	decimal(10,2)		Yes	NULL				Change Drop More
<input type="checkbox"/>	4 new_price	decimal(10,2)		Yes	NULL				Change Drop More
<input type="checkbox"/>	5 old_quantity	int(11)		Yes	NULL				Change Drop More
<input type="checkbox"/>	6 new_quantity	int(11)		Yes	NULL				Change Drop More
<input type="checkbox"/>	7 timestamp	timestamp		No		current_timestamp()			Change Drop More

Run SQL query/queries on table bakery.product_audit: [?](#)

```

1 DELIMITER //
2
3 CREATE TRIGGER after_products_update_2
4 AFTER UPDATE ON products
5 FOR EACH ROW
6 BEGIN
7     INSERT INTO product_audit (product_id, old_price, new_price, old_quantity, new_quantity)
8     VALUES (OLD.product_id, OLD.price, NEW.price, OLD.quantity, NEW.quantity);
9 END //
10
11 DELIMITER ;
12

```

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: [?](#)

```

1 UPDATE products
2 SET price = 1000.00, quantity = 100
3 WHERE product_id = 1;
4

```

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 [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

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

Show all | Number of rows: 25 Search this table

[Extra options](#)

[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
<input type="checkbox"/>	1	Croissant	1000.00	100	Classic Corissant, Chocolate Corissant, Cheese Cor...
<input type="checkbox"/>	2	Cookies	20.00	40	Classic Cookies, Pistachio Cookies, Kinder Cookies
<input type="checkbox"/>	3	Pizza	25.00	20	Pepperoni Pizza, Margherite Pizza, Ranch Pizza
<input type="checkbox"/>	4	Muffin	15.00	50	Blueberry Muffin, Chocolate Muffin, Vanilla Muffin
<input type="checkbox"/>	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
<input type="checkbox"/>	2	Cookies	20.00	40	Classic Cookies, Pistachio Cookies, Kinder Cookies
<input type="checkbox"/>	3	Pizza	25.00	20	Pepperoni Pizza, Margherite Pizza, Ranch Pizza
<input type="checkbox"/>	4	Muffin	15.00	50	Blueberry Muffin, Chocolate Muffin, Vanilla Muffin
<input type="checkbox"/>	5	Beard	10.00	100	White Bread, Bagel, Pita Bread

← ↑ Check all With selected: Edit Copy Delete Export