REPORT
DATABASE PROGRAMMING FINAL PROJECT
skincarestore.sql

Lecturer: Novi Prisma Yunita M.Kom

Group Member:

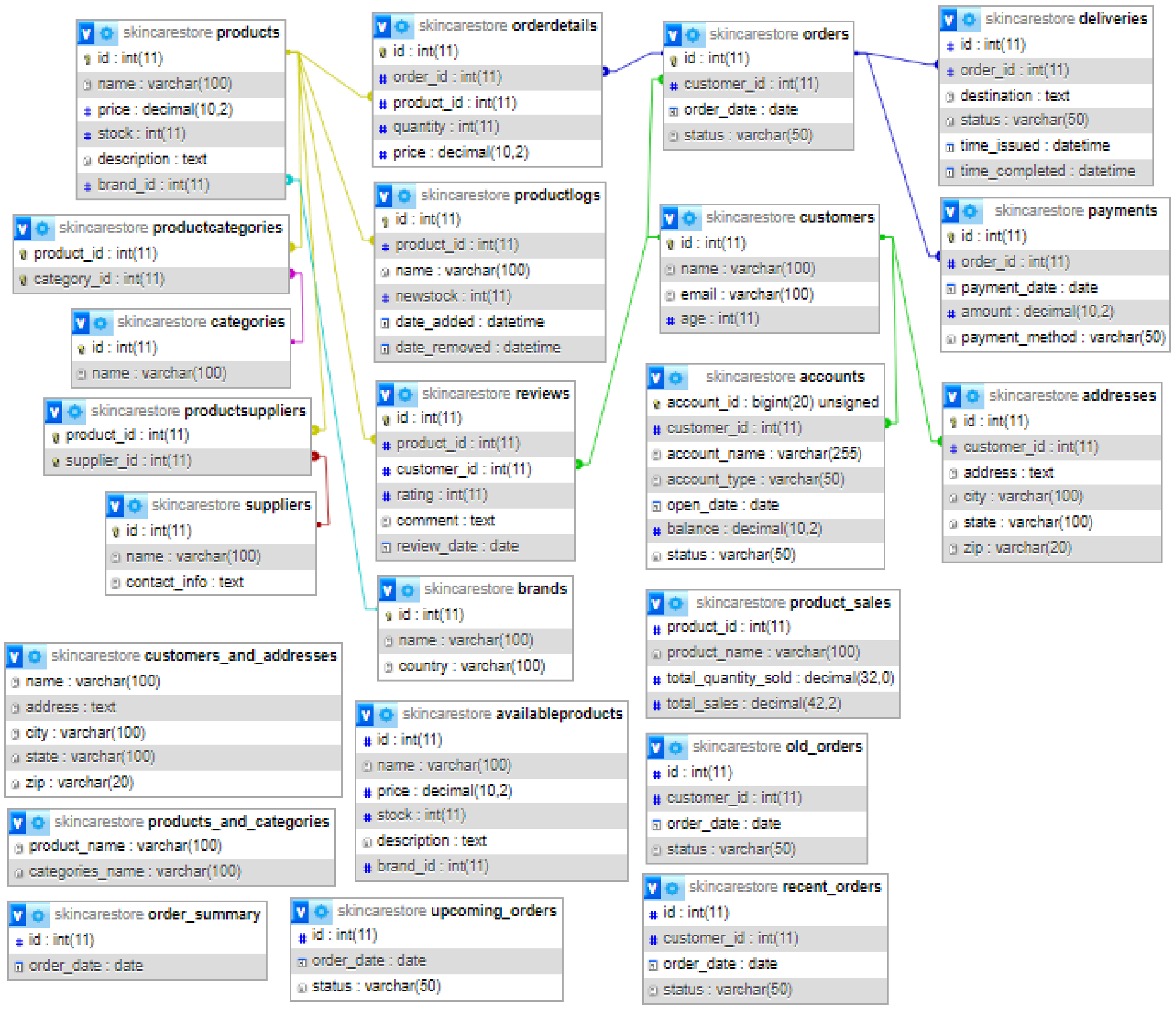
1.	Muhammad Fabian Nurdin	22.61.0233
2.	Stephen Darma Putra N.	22.61.0238
3.	Muhammad Dawam Amali	22.61.0239
4.	Nada Satya Maharani	22.61.0240

STUDY PROGRAM S1 INFORMATICS, FACULTY OF COMPUTER SCIENCE

AMIKOM UNIVERSITY YOGYAKARTA

2023

ERD



The database that our group created is the database for the skincare store system. The following is an explanation of the main structure of the database;

GITHUB

https://github.com/arutaruumu/SKINCARE-STORE-DB

Main table

1. **Customers** : Store customer data
   - Columns : ‘id’, ‘name’, ‘email’, ‘age’
2. **Brands** : Store skincare brand data

    - Columns : ‘id’, ‘name’, ‘country’
3. **Products** : Store data on skincare products sold
    - Columns : ‘id’, ‘name’, ‘price’, ‘stock’, ‘description’, ‘brand_id’
4. **Orders** : Stores order data placed by customers
    - Columns : ‘id’, ‘customer_id’, ‘order_date’, ‘status’
5. **OrderDetails** : Stores details of each order
    - Columns : ‘id’, ‘order_id’, ‘product_id’, ‘quantity’, ‘price’
6. **Categories** : Stores skincare product category data
    - Columns : ‘id’, ‘name’
7. **ProductCategories** : Bridge table for many-to-many relationship between products and categories
    - Columns : ‘product_id’, ‘category_id’
8. **Reviews** : Stores customer reviews on skincare products
    - Columns : ‘id’, ‘product_id’, ‘customer_id’, ‘rating’, ‘comment’, ‘review_date’
9. **Addresses** : Stores customer address data
    - Columns : ‘id’, ‘customer_id’, ‘address’, ‘city’, ‘state’, ‘zip’
10. **Payments** : Stores payment data for each order
    - Columns : ‘id’, ‘order_id’, ‘payment_date’, ‘amount’, ‘payment_method’
11. **Suppliers** : Stores data of skincare product suppliers
    - Columns : ‘id’, ‘name’, ‘contact_info’
12. **ProductSuppliers** : Bridge table for many-to-many relationship between products and suppliers
    - Columns : ‘product_id’, ‘supplier_id’

Relationship between tables

1. **Customers to orders: one-to-many (one customer can place multiple orders)**
2. **Orders to OrderDetails: one-to-many (one order can have many order details)**
3. **Products to OrderDetails: one-to-many (one product can appear in many order details)**
4. **Products to Categories through ProductCategories: many-to-many (many products can belong to many categories)**
5. **Brands to Products: one-to-many (one brand has many products)**
6. **Products to reviews: one-to-many (one product can have many reviews)**
7. **Customers to reviews: one-to-many (one customer can give many reviews)**
8. **Customers to Addresses: one-to-many (one customer can have many addresses)**
9. **Orders to payments: one-to-many (one order can have many payments)**
10. **Suppliers to Products through ProductSuppliers: many-to-many (many suppliers can supply many products)**

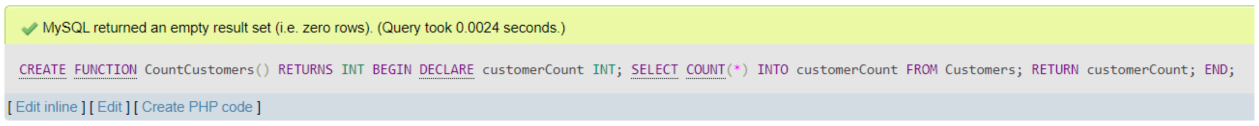
Function

Final Project skincarestore.sql | Create Function with multiple models, and Execute function

1. Create 1 function, 1 function with empty parameters

Jawab:

```
DELIMITER //

CREATE FUNCTION CountCustomers() RETURNS INT
BEGIN
    DECLARE customerCount INT;
    SELECT COUNT(*) INTO customerCount FROM Customers;
    RETURN customerCount;
END //

DELIMITER ;
```



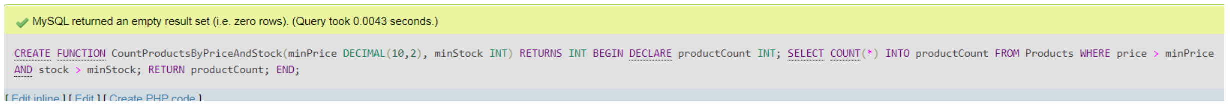
```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0024 seconds.)

CREATE FUNCTION CountCustomers() RETURNS INT BEGIN DECLARE customerCount INT; SELECT COUNT(*) INTO customerCount FROM Customers; RETURN customerCount; END;
```

Explanation: Create one function to calculate the total number of customers in the customers table.

2. Create 1 function, with 2 parameters.

Jawab:

```
DELIMITER //
CREATE FUNCTION CountProductsByPriceAndStock(minPrice
DECIMAL(10,2), minStock INT)
RETURNS INT
BEGIN
DECLARE productCount INT;
SELECT COUNT(*) INTO productCount
FROM Products
WHERE price > minPrice AND stock > minStock; RETURN productCount;
END //
DELIMITER ;
```



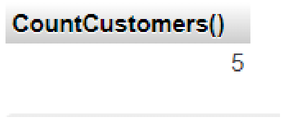
```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0043 seconds.)

CREATE FUNCTION CountProductsByPriceAndStock(minPrice DECIMAL(10,2), minStock INT) RETURNS INT BEGIN DECLARE productCount INT; SELECT COUNT(*) INTO productCount FROM Products WHERE price > minPrice
AND stock > minStock; RETURN productCount; END;
```

Explanation: Counts the number of products that have more than a certain price and more than a certain amount of stock.

3. Execution of each function

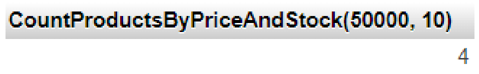
Jawab:

```
SELECT CountCustomers();
```



```
CountCustomers()
               5
```

```
SELECT CountProductsByPriceAndStock(50000, 10);
```



```
CountProductsByPriceAndStock(50000, 10)
                                      4
```

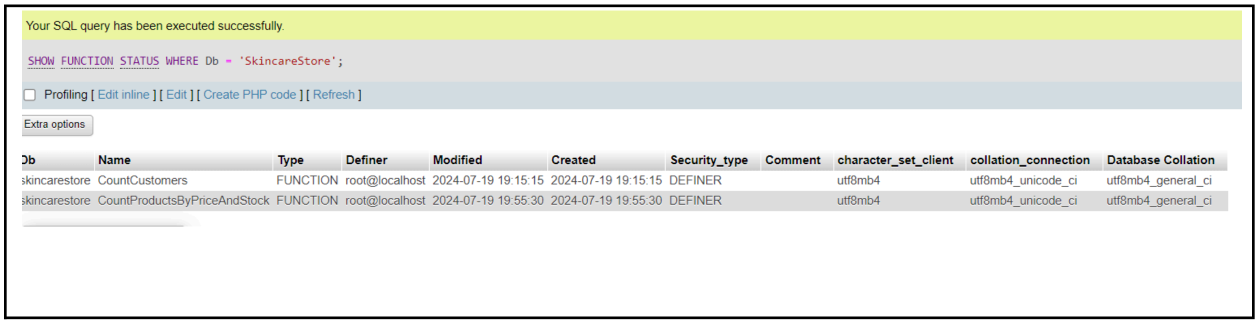
Explanation:

- The first function aims to calculate the total customers in the customer product, where there are only 5 customers in the customers table.
- The second function aims to count the number of products that have a price of more than a certain price and more than a certain amount of stock, which is a product with a price of 50,000 with a stock of 10, there are 4 products.

4. Display the list of functions

Jawab:

```
SHOW FUNCTION STATUS WHERE Db = 'SkincareStore';
```



Your SQL query has been executed successfully.

```
SHOW FUNCTION STATUS WHERE Db = 'SkincareStore';
```

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

Extra options

Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_client	collation_connection	Database Collation
skincarestore	CountCustomers	FUNCTION	root@localhost	2024-07-19 19:15:15	2024-07-19 19:15:15	DEFINER		utf8mb4	utf8mb4_unicode_ci	utf8mb4_general_ci
skincarestore	CountProductsByPriceAndStock	FUNCTION	root@localhost	2024-07-19 19:55:30	2024-07-19 19:55:30	DEFINER		utf8mb4	utf8mb4_unicode_ci	utf8mb4_general_ci

Explanation: Displays all functions in the database

Procedure

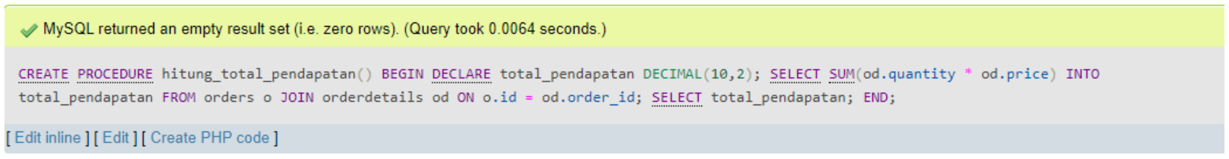
Final Project skincarestore.sql | Create Procedure and Execute function

1. Create 2 procedures, 1 procedure with empty parameters

Calculate total income

Jawab:

```
DELIMITER $$
CREATE PROCEDURE hitung_total_pendapatan()
BEGIN
    DECLARE total_pendapatan DECIMAL(10,2);
    SELECT SUM(od.quantity * od.price) INTO total_pendapatan
    FROM orders o
    JOIN orderdetails od ON o.id = od.order_id;
    SELECT total_pendapatan;
END $$
DELIMITER ;
```

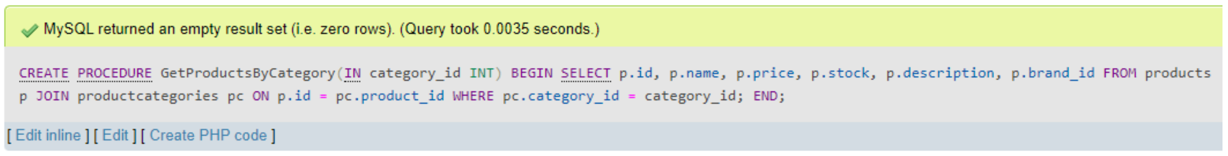


```
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0064 seconds.)

CREATE PROCEDURE hitung_total_pendapatan() BEGIN DECLARE total_pendapatan DECIMAL(10,2); SELECT SUM(od.quantity * od.price) INTO
total_pendapatan FROM orders o JOIN orderdetails od ON o.id = od.order_id; SELECT total_pendapatan; END;
```

View products according to category

Jawab:

```
DELIMITER //
CREATE PROCEDURE GetProductsByCategory(IN category_id INT)
BEGIN
    SELECT p.id, p.name, p.price, p.stock, p.description, p.brand_id
    FROM products p
    JOIN productcategories pc ON p.id = pc.product_id
    WHERE pc.category_id = category_id;
END //
DELIMITER ;
```



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0035 seconds.)

```
CREATE PROCEDURE GetProductsByCategory(IN category_id INT) BEGIN SELECT p.id, p.name, p.price, p.stock, p.description, p.brand_id FROM products
p JOIN productcategories pc ON p.id = pc.product_id WHERE pc.category_id = category_id; END;
```

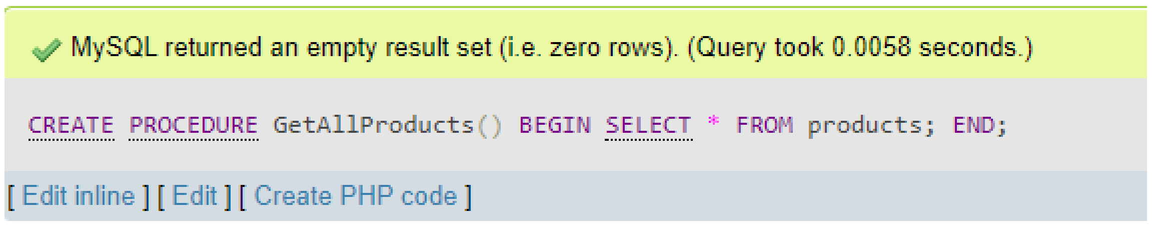
[Edit inline] [Edit] [Create PHP code]

2. Create 2 procedures, 1 procedure with 2 parameters

Display all products

Jawab:

```
DELIMITER //
CREATE PROCEDURE GetAllProducts()
BEGIN
    SELECT * FROM products;
END //
DELIMITER ;
```



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0058 seconds.)

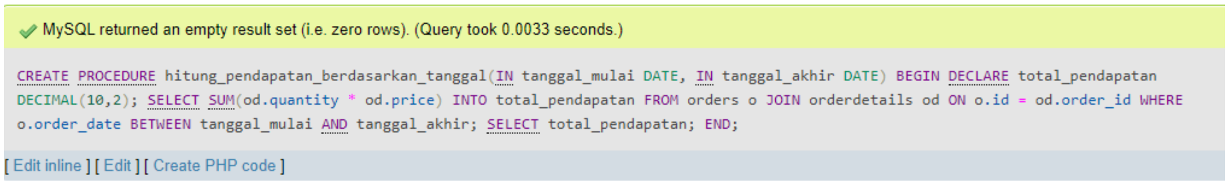
```
CREATE PROCEDURE GetAllProducts() BEGIN SELECT * FROM products; END;
```

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

Generate total revenue over a period of time

Jawab:

```
DELIMITER $$

CREATE PROCEDURE hitung_pendapatan_berdasarkan_tanggal(IN tanggal_mulai DATE,
IN tanggal_akhir DATE)

BEGIN

    DECLARE total_pendapatan DECIMAL(10,2);

    SELECT SUM(od.quantity * od.price) INTO total_pendapatan

    FROM orders o

    JOIN orderdetails od ON o.id = od.order_id

    WHERE o.order_date BETWEEN tanggal_mulai AND tanggal_akhir;

    SELECT total_pendapatan;

END $$

DELIMITER ;
```



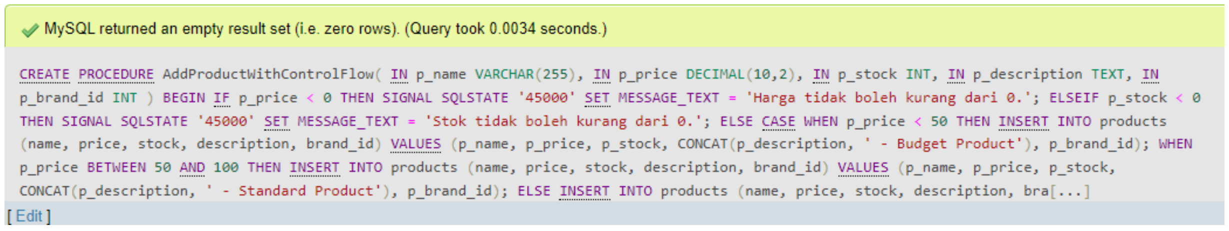
```
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0033 seconds.)

CREATE PROCEDURE hitung_pendapatan_berdasarkan_tanggal(IN tanggal_mulai DATE, IN tanggal_akhir DATE) BEGIN DECLARE total_pendapatan DECIMAL(10,2); SELECT SUM(od.quantity * od.price) INTO total_pendapatan FROM orders o JOIN orderdetails od ON o.id = od.order_id WHERE o.order_date BETWEEN tanggal_mulai AND tanggal_akhir; SELECT total_pendapatan; END;
```

3. Prosedur penyimpanan mengandung aliran kontrol (pernyataan IF, CASE, atau LOOP)

Answer:

```
DELIMITER $$

CREATE PROCEDURE AddProductWithControlFlow(

    IN p_name VARCHAR(255),

    IN p_price DECIMAL(10,2),

    IN p_stock INT,

    IN p_description TEXT,

    IN p_brand_id INT

)

BEGIN
```

```
   IF p_price < 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Harga tidak boleh kurang dari 0.';
   ELSEIF p_stock < 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Stok tidak boleh kurang dari 0.';
   ELSE
      CASE
         WHEN p_price < 50 THEN
            INSERT INTO products (name, price, stock, description, brand_id)
            VALUES (p_name, p_price, p_stock, CONCAT(p_description, ' - Budget Product'),
p_brand_id);
         WHEN p_price BETWEEN 50 AND 100 THEN
            INSERT INTO products (name, price, stock, description, brand_id)
            VALUES (p_name, p_price, p_stock, CONCAT(p_description, ' - Standard Product'),
p_brand_id);
         ELSE
            INSERT INTO products (name, price, stock, description, brand_id)
            VALUES (p_name, p_price, p_stock, CONCAT(p_description, ' - Premium Product'),
p_brand_id);
      END CASE;
   END IF;
END$$
DELIMITER ;
```

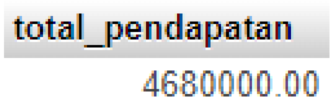


```
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0034 seconds.)

CREATE PROCEDURE AddProductWithControlFlow( IN p_name VARCHAR(255), IN p_price DECIMAL(10,2), IN p_stock INT, IN p_description TEXT, IN
p_brand_id INT ) BEGIN IF p_price < 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Harga tidak boleh kurang dari 0.'; ELSEIF p_stock < 0
THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Stok tidak boleh kurang dari 0.'; ELSE CASE WHEN p_price < 50 THEN INSERT INTO products
(name, price, stock, description, brand_id) VALUES (p_name, p_price, p_stock, CONCAT(p_description, ' - Budget Product'), p_brand_id); WHEN
p_price BETWEEN 50 AND 100 THEN INSERT INTO products (name, price, stock, description, brand_id) VALUES (p_name, p_price, p_stock,
CONCAT(p_description, ' - Standard Product'), p_brand_id); ELSE INSERT INTO products (name, price, stock, description, bra[...]
[ Edit ]
```

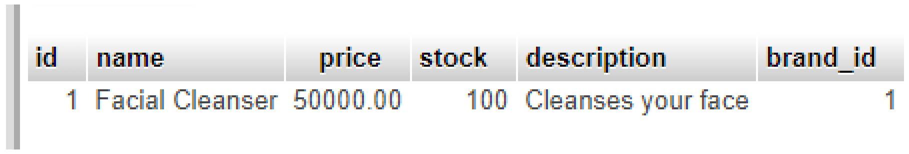
4. Execute each procedure

Answer:

1. Calculating total revenue

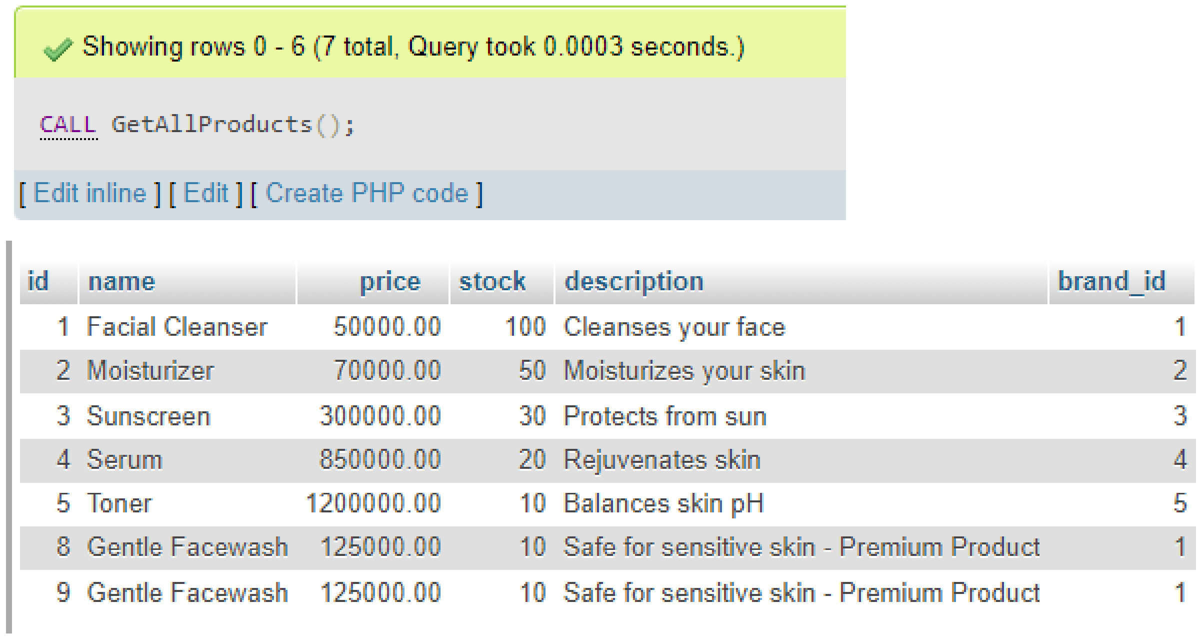
`CALL hitung_total_pendapatan();`
 total_pendapatan 4680000.00
Calculate all revenue from sales

2. View products according to category

CALL GetProductsByCategory(1)

Display products with category_id = 1

id	name	price	stock	description	brand_id
1	Facial Cleanser	50000.00	100	Cleanses your face	1

3. Display all existing products

CALL GetAllProducts()

Calculate revenue according to the date inputted

✓ Showing rows 0 - 6 (7 total, Query took 0.0003 seconds.)

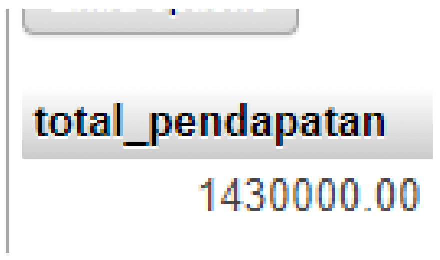
```
CALL GetAllProducts();
```

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

id	name	price	stock	description	brand_id
1	Facial Cleanser	50000.00	100	Cleanses your face	1
2	Moisturizer	70000.00	50	Moisturizes your skin	2
3	Sunscreen	300000.00	30	Protects from sun	3
4	Serum	850000.00	20	Rejuvenates skin	4
5	Toner	1200000.00	10	Balances skin pH	5
8	Gentle Facewash	125000.00	10	Safe for sensitive skin - Premium Product	1
9	Gentle Facewash	125000.00	10	Safe for sensitive skin - Premium Product	1

4. Calculating total income over time

```
CALL hitung_pendapatan_berdasarkan_tanggal('2024-07-01', '2024-07-03');
```

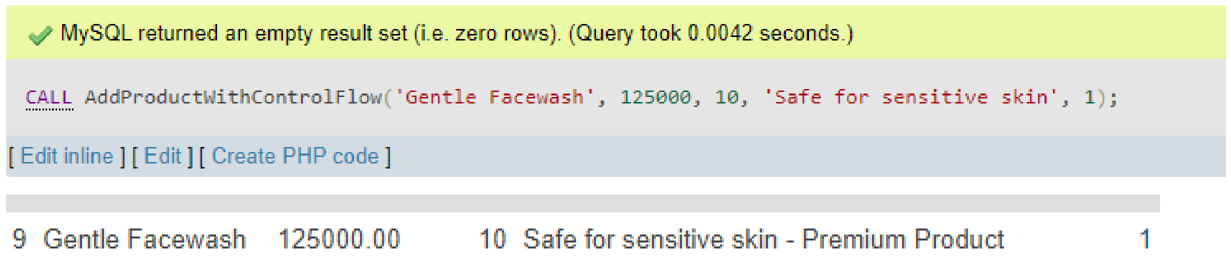


```
total_pendapatan
      1430000.00
```

Calculate revenue according to the date inputted

5. Adding new products

```
CALL AddProductWithControlFlow('Gentle Facewash', 125000, 10, 'Safe for sensitive skin', 1);
```

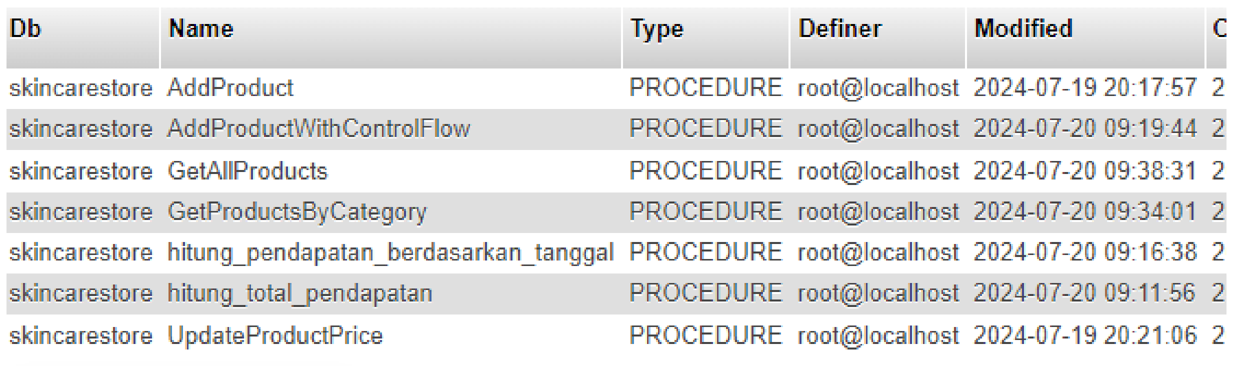


```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0042 seconds.)

CALL AddProductWithControlFlow('Gentle Facewash', 125000, 10, 'Safe for sensitive skin', 1);

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

9  Gentle Facewash    125000.00        10  Safe for sensitive skin - Premium Product            1
```

Added new products with control flow.

5. Show Procedure List

Answer:



Db	Name	Type	Definer	Modified	C
skincarestore	AddProduct	PROCEDURE	root@localhost	2024-07-19 20:17:57	2
skincarestore	AddProductWithControlFlow	PROCEDURE	root@localhost	2024-07-20 09:19:44	2
skincarestore	GetAllProducts	PROCEDURE	root@localhost	2024-07-20 09:38:31	2
skincarestore	GetProductsByCategory	PROCEDURE	root@localhost	2024-07-20 09:34:01	2
skincarestore	hitung_pendapatan_berdasarkan_tanggal	PROCEDURE	root@localhost	2024-07-20 09:16:38	2
skincarestore	hitung_total_pendapatan	PROCEDURE	root@localhost	2024-07-20 09:11:56	2
skincarestore	UpdateProductPrice	PROCEDURE	root@localhost	2024-07-19 20:21:06	2

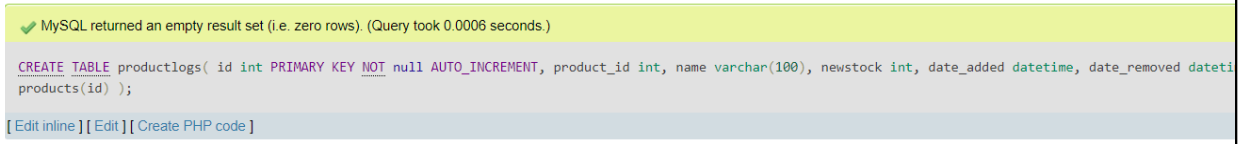
TRIGGER

Final Project skincarestore.sql | Membuat Trigger, dan Mengeksekusi Trigger.

1. Create several log tables to store data from trigger execution

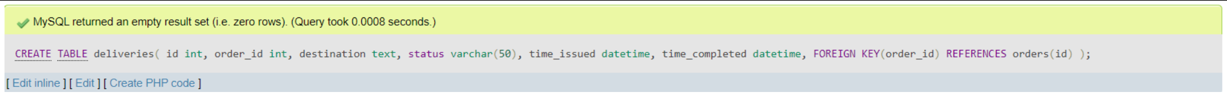
Jawab:

```
CREATE TABLE productlogs(
    id int PRIMARY KEY NOT null AUTO_INCREMENT,
    product_id int,
    name varchar(100),
     newstock int,
    date_added datetime,
    date_removed datetime,
    FOREIGN KEY(product_id) REFERENCES products(id)
    )
```



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

CREATE TABLE productlogs( id int PRIMARY KEY NOT null AUTO_INCREMENT, product_id int, name varchar(100), newstock int, date_added datetime, date_removed dateti
products(id) );

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

```
CREATE TABLE deliveries(
    id int,
    order_id int,
    destination text,
    status varchar(50),
    time_issued datetime,
    time_completed datetime,
    FOREIGN KEY(order_id) REFERENCES orders(id)
    )
```



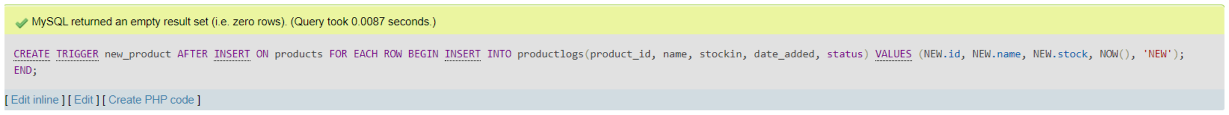
```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0008 seconds.)

CREATE TABLE deliveries( id int, order_id int, destination text, status varchar(50), time_issued datetime, time_completed datetime, FOREIGN KEY(order_id) REFERENCES orders(id) );

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

2. Create 6 triggers, consisting of BEFORE and AFTER

-

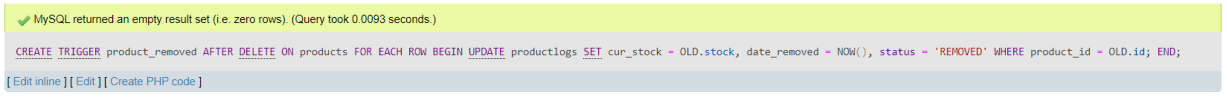
Jawab:

```
# AFTER INSERT
DELIMITER //
CREATE TRIGGER new_product
AFTER INSERT
ON products FOR EACH ROW
BEGIN
    INSERT INTO productlogs(product_id, name, stockin, date_added,
status) VALUES (NEW.id, NEW.name, NEW.stock, NOW(), 'NEW');
END //
```



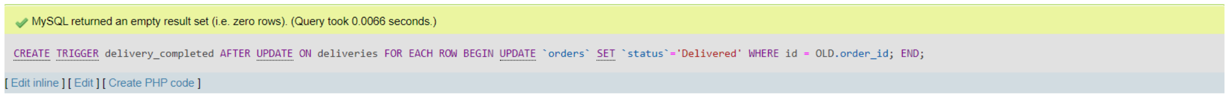
```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0087 seconds.)

CREATE TRIGGER new_product AFTER INSERT ON products FOR EACH ROW BEGIN INSERT INTO productlogs(product_id, name, stockin, date_added, status) VALUES (NEW.id, NEW.name, NEW.stock, NOW(), 'NEW');
END;

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

```
# AFTER DELETE
DELIMITER //
CREATE TRIGGER product_removed
AFTER DELETE
ON products FOR EACH ROW
BEGIN
     UPDATE productlogs SET cur_stock = OLD.stock, date_removed =
NOW(), status = 'REMOVED' WHERE product_id = OLD.id;
END //
```



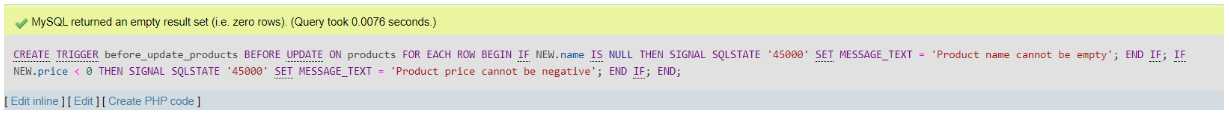
```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0093 seconds.)

CREATE TRIGGER product_removed AFTER DELETE ON products FOR EACH ROW BEGIN UPDATE productlogs SET cur_stock = OLD.stock, date_removed = NOW(), status = 'REMOVED' WHERE product_id = OLD.id; END;

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

```
# AFTER UPDATE

DELIMITER //

CREATE TRIGGER delivery_completed

AFTER UPDATE

ON deliveries FOR EACH ROW

BEGIN

     UPDATE `orders` SET `status`='Delivered' WHERE id =
OLD.order_id;

END //
```



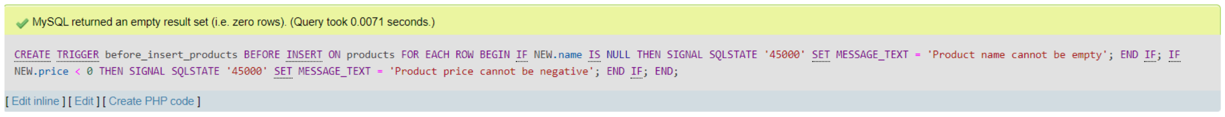
```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0066 seconds.)

CREATE TRIGGER delivery_completed AFTER UPDATE ON deliveries FOR EACH ROW BEGIN UPDATE `orders` SET `status`='Delivered' WHERE id = OLD.order_id; END;

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

```
# BEFORE UPDATE

DELIMITER //

CREATE TRIGGER before_update_products

BEFORE UPDATE

ON products FOR EACH ROW

BEGIN

    IF NEW.name IS NULL THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product name
cannot be empty';

    END IF;


    IF NEW.price < 0 THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product price
cannot be negative';

    END IF;

END //
```



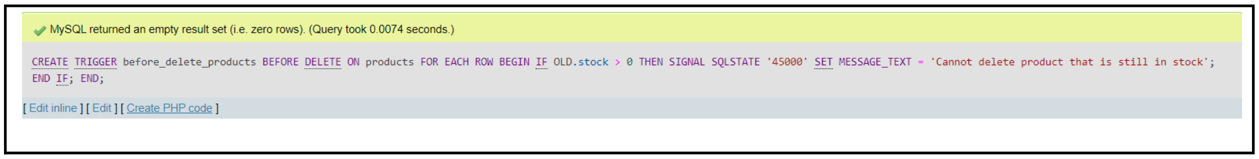
```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0076 seconds.)

CREATE TRIGGER before_update_products BEFORE UPDATE ON products FOR EACH ROW BEGIN IF NEW.name IS NULL THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product name cannot be empty'; END IF; IF
NEW.price < 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product price cannot be negative'; END IF; END;

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

```
# BEFORE INSERT

DELIMITER //

CREATE TRIGGER before_insert_products

BEFORE INSERT

ON products FOR EACH ROW

BEGIN

    IF NEW.name IS NULL THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product name
cannot be empty';

    END IF;


    IF NEW.price < 0 THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product price
cannot be negative';

    END IF;

END //
```



```
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0071 seconds.)

CREATE TRIGGER before_insert_products BEFORE INSERT ON products FOR EACH ROW BEGIN IF NEW.name IS NULL THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product name cannot be empty'; END IF; IF
NEW.price < 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Product price cannot be negative'; END IF; END;
```

```
# BEFORE DELETE

DELIMITER //

CREATE TRIGGER before_delete_products

BEFORE DELETE

ON products FOR EACH ROW

BEGIN

    IF OLD.stock > 0 THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot delete
product that is still in stock';

    END IF;

END //
```



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0074 seconds.)

CREATE TRIGGER before_delete_products BEFORE DELETE ON products FOR EACH ROW BEGIN IF OLD.stock > 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot delete product that is still in stock';
END IF; END;
```

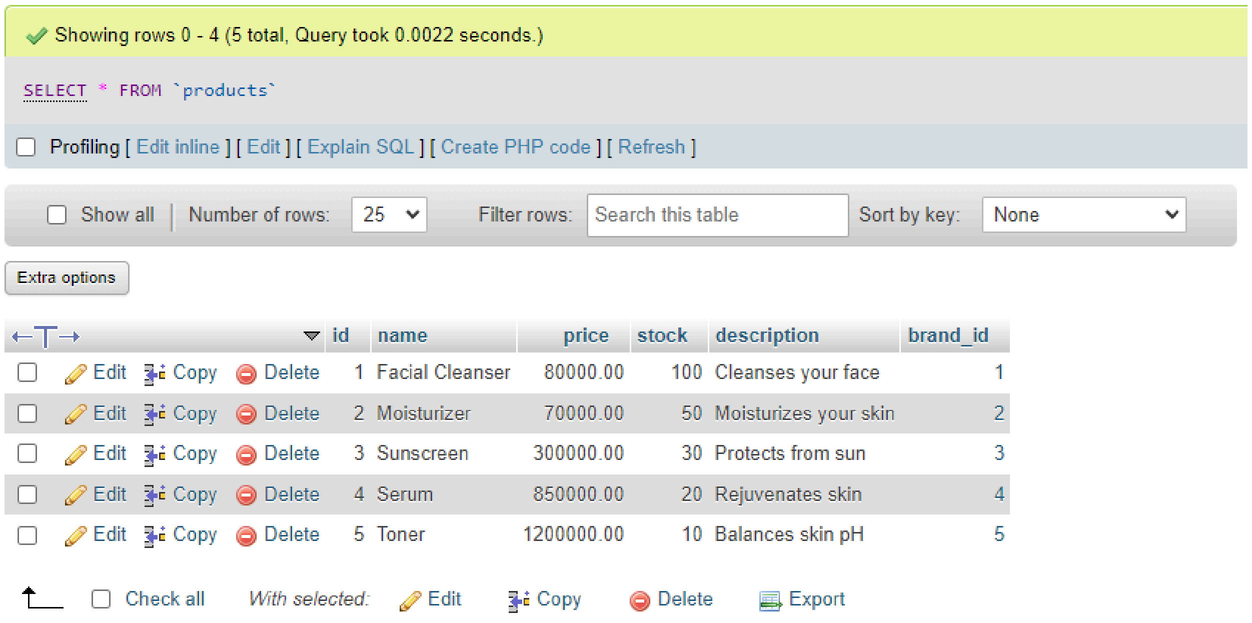
membuat 6 trigger dengan nama

- ***new_product (after insert)***
    - this trigger is run after a new product is inserted to the *products* table, with target table *productlogs*.
- ***product_removed (after delete)***
    - this trigger is run after a product is deleted from the *products* table, with target table *productlogs*.
- ***delivery_completed (after update)***
    - this trigger is run after the status of a delivery in table *deliveries* is changed into “completed” with target table *orders.*
- ***before_update_products (before update)***
    - this trigger is run before a product is edited in the *products* table, to validate whether the edited product has a valid name and/or price.
- ***before_insert_products (before insert)***
    - this trigger is run before a product is inserted in the *products* table, to validate whether the inserted product has a valid name and/or price.
- ***before_delete_products (before delete)***
    - this trigger is run before a product is deleted in the *products* table, to validate whether the deleted product has a valid stock count.

3. Execute each EVENT and record them to the corresponding table

- **INSERT**

SEBELUM



✓ Showing rows 0 - 4 (5 total, Query took 0.0022 seconds.)

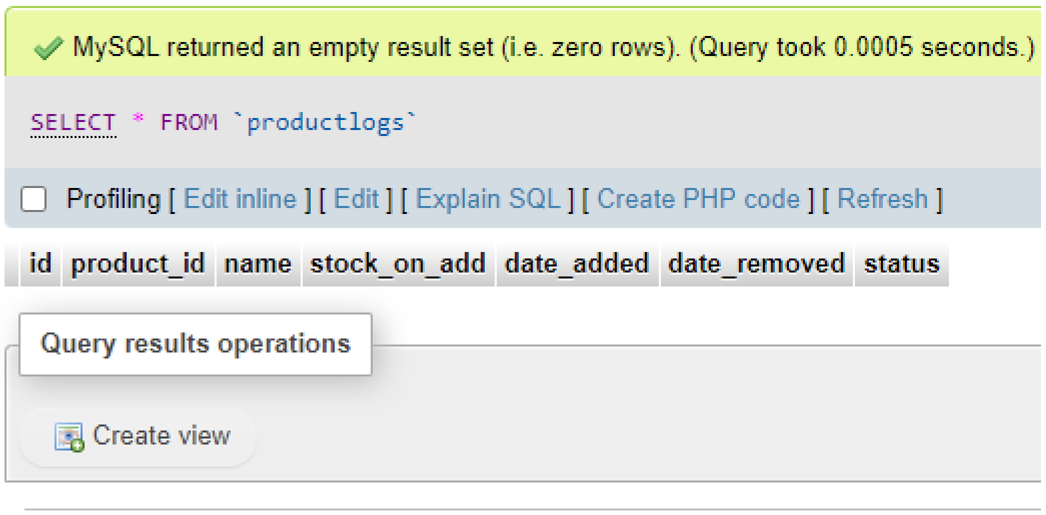
```
SELECT * FROM `products`
```

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

id	name	price	stock	description	brand_id
1	Facial Cleanser	80000.00	100	Cleanses your face	1
2	Moisturizer	70000.00	50	Moisturizes your skin	2
3	Sunscreen	300000.00	30	Protects from sun	3
4	Serum	850000.00	20	Rejuvenates skin	4
5	Toner	1200000.00	10	Balances skin pH	5



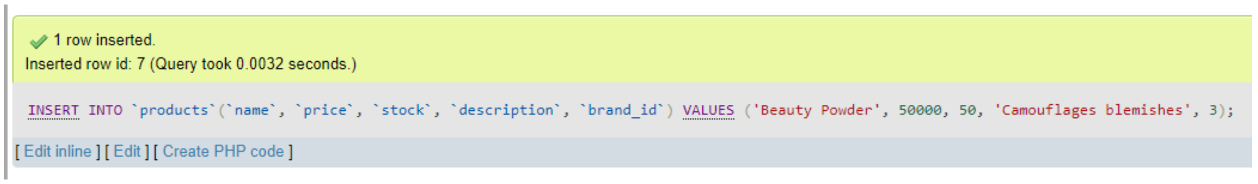
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

```
SELECT * FROM `productlogs`
```

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

id	product_id	name	stock_on_add	date_added	date_removed	status

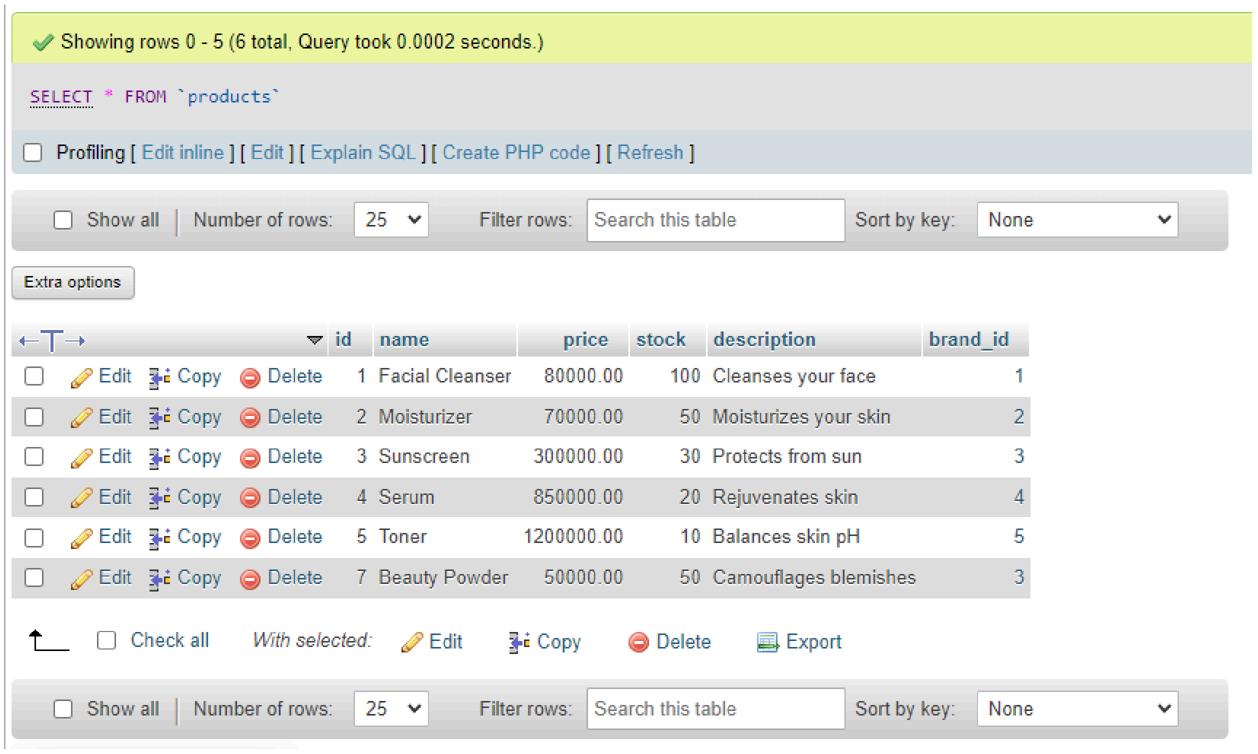
SETELAH INSERT



✓ 1 row inserted.
Inserted row id: 7 (Query took 0.0032 seconds.)

```
INSERT INTO `products`(`name`, `price`, `stock`, `description`, `brand_id`) VALUES ('Beauty Powder', 50000, 50, 'Camouflages blemishes', 3);
```

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



✓ Showing rows 0 - 5 (6 total, Query took 0.0002 seconds.)

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

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

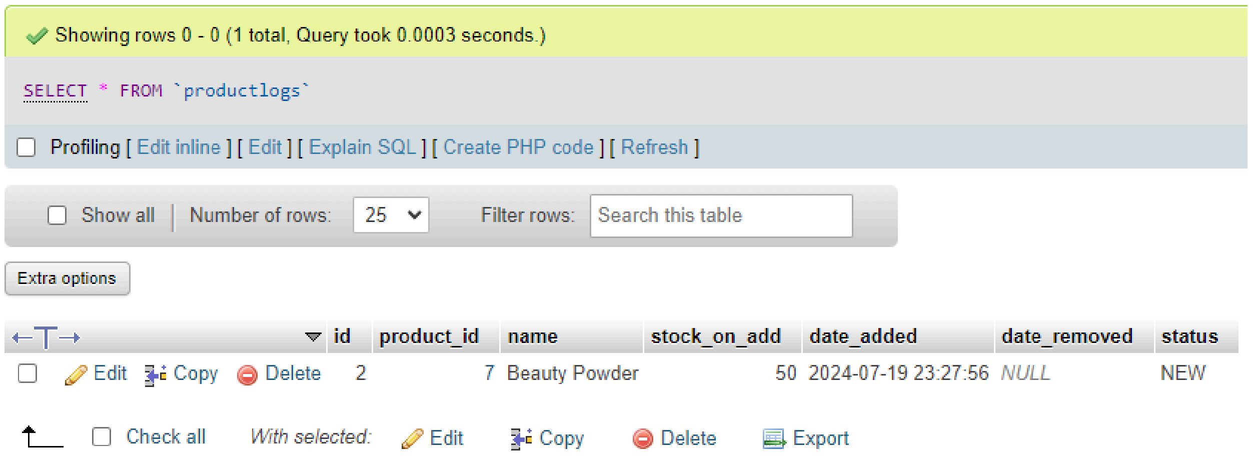
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

id	name	price	stock	description	brand_id
1	Facial Cleanser	80000.00	100	Cleanses your face	1
2	Moisturizer	70000.00	50	Moisturizes your skin	2
3	Sunscreen	300000.00	30	Protects from sun	3
4	Serum	850000.00	20	Rejuvenates skin	4
5	Toner	1200000.00	10	Balances skin pH	5
7	Beauty Powder	50000.00	50	Camouflages blemishes	3

Check all With selected: Edit Copy Delete Export

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



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

```
SELECT * FROM `productlogs`
```

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

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

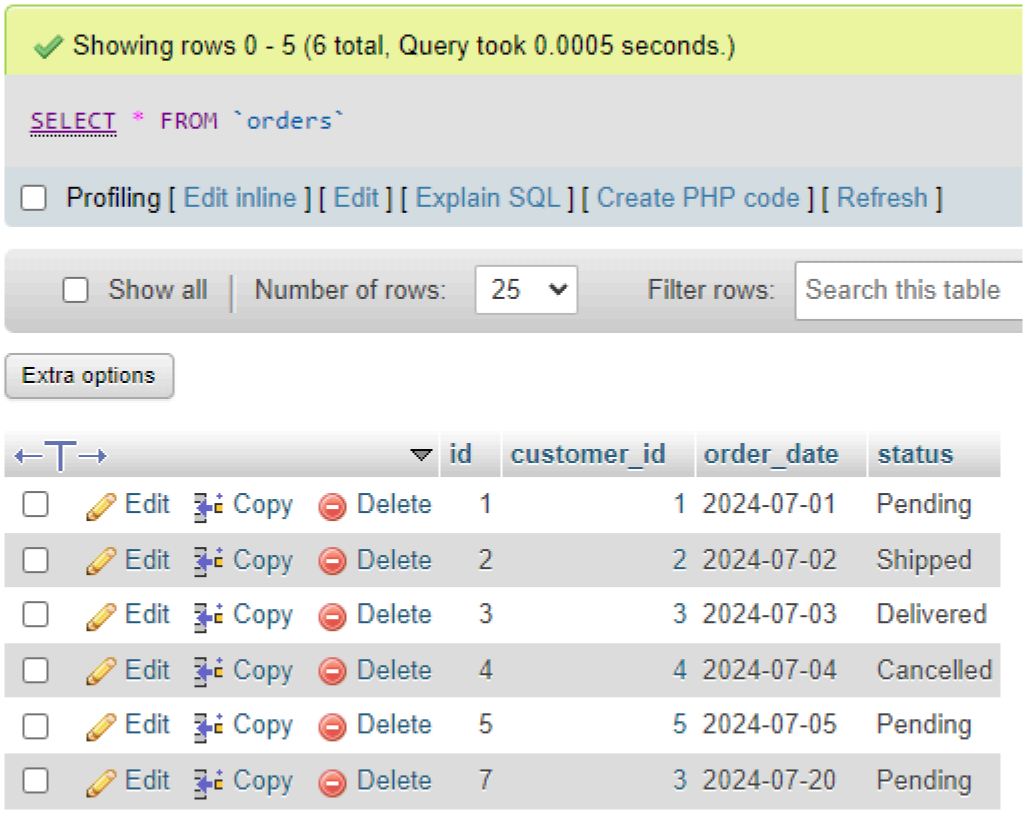
Extra options

id	product_id	name	stock_on_add	date_added	date_removed	status
2	7	Beauty Powder	50	2024-07-19 23:27:56	*NULL*	NEW

Check all With selected: Edit Copy Delete Export

- **UPDATE**

SEBELUM



✓ Showing rows 0 - 5 (6 total, Query took 0.0005 seconds.)

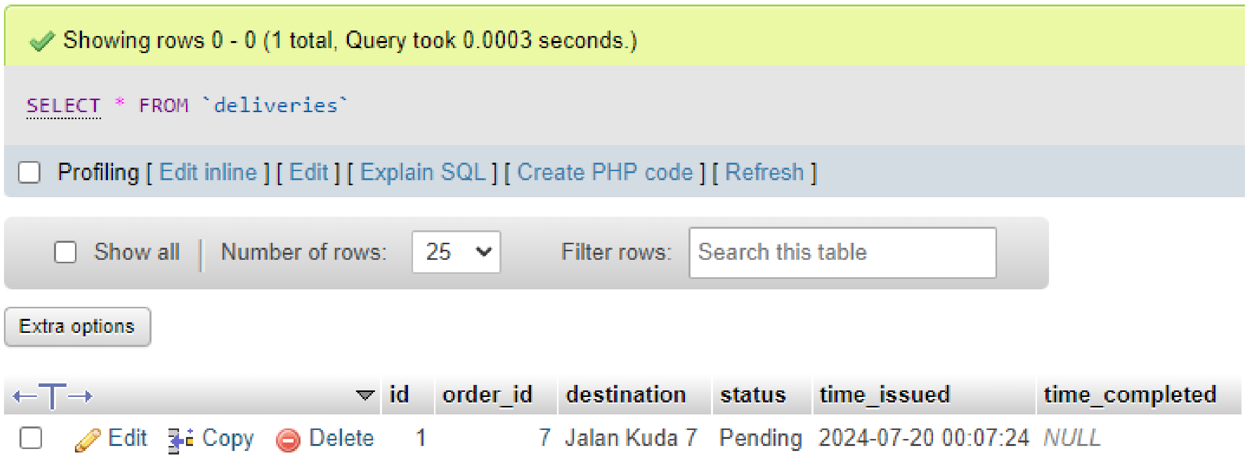
```
SELECT * FROM `orders`
```

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

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

Extra options

	id	customer_id	order_date	status
Edit Copy Delete	1	1	2024-07-01	Pending
Edit Copy Delete	2	2	2024-07-02	Shipped
Edit Copy Delete	3	3	2024-07-03	Delivered
Edit Copy Delete	4	4	2024-07-04	Cancelled
Edit Copy Delete	5	5	2024-07-05	Pending
Edit Copy Delete	7	3	2024-07-20	Pending



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

```
SELECT * FROM `deliveries`
```

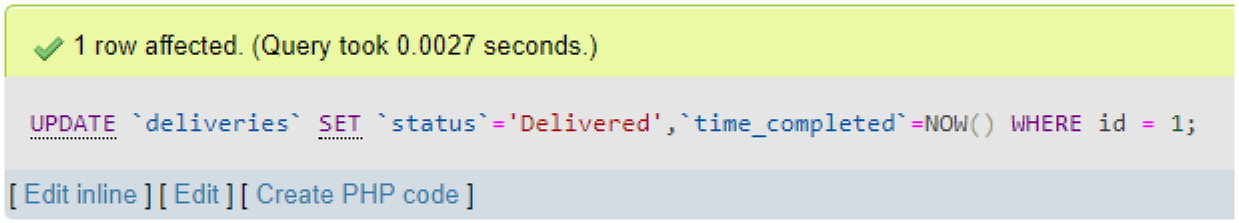
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

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

Extra options

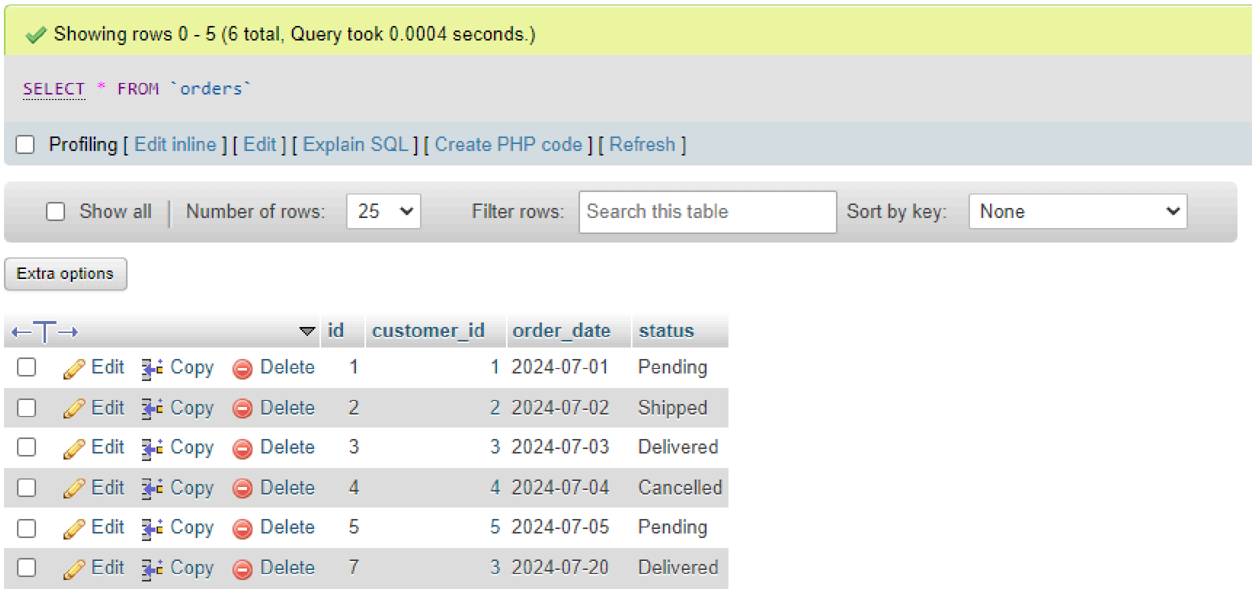
	id	order_id	destination	status	time_issued	time_completed
Edit Copy Delete	1	7	Jalan Kuda 7	Pending	2024-07-20 00:07:24	NULL

SETELAH UPDATE



1 row affected. (Query took 0.0027 seconds.)

```
UPDATE `deliveries` SET `status`='Delivered',`time_completed`=NOW() WHERE id = 1;
```

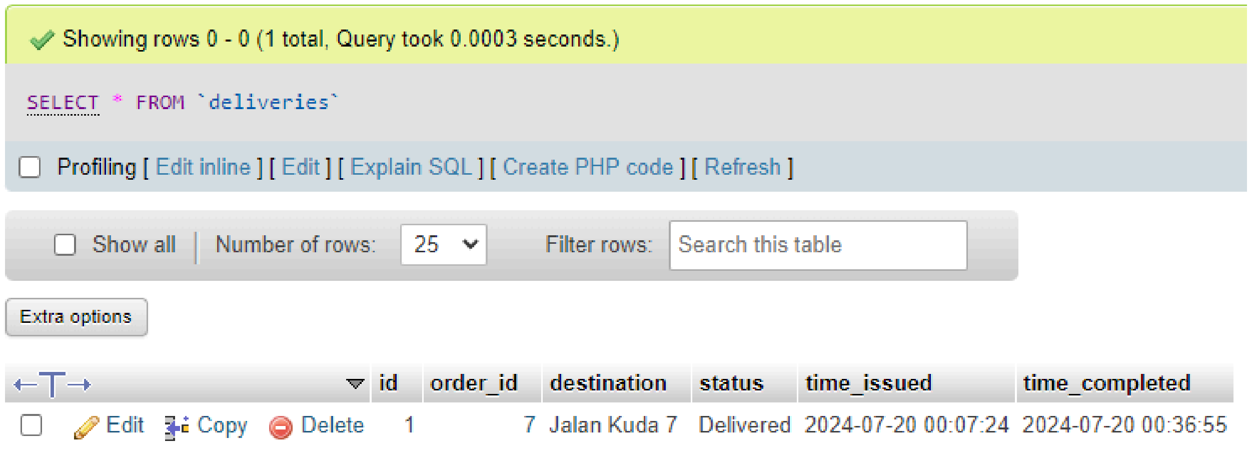


Showing rows 0 - 5 (6 total, Query took 0.0004 seconds.)

```
SELECT * FROM `orders`
```

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

id	customer_id	order_date	status
1	1	2024-07-01	Pending
2	2	2024-07-02	Shipped
3	3	2024-07-03	Delivered
4	4	2024-07-04	Cancelled
5	5	2024-07-05	Pending
7	3	2024-07-20	Delivered



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

```
SELECT * FROM `deliveries`
```

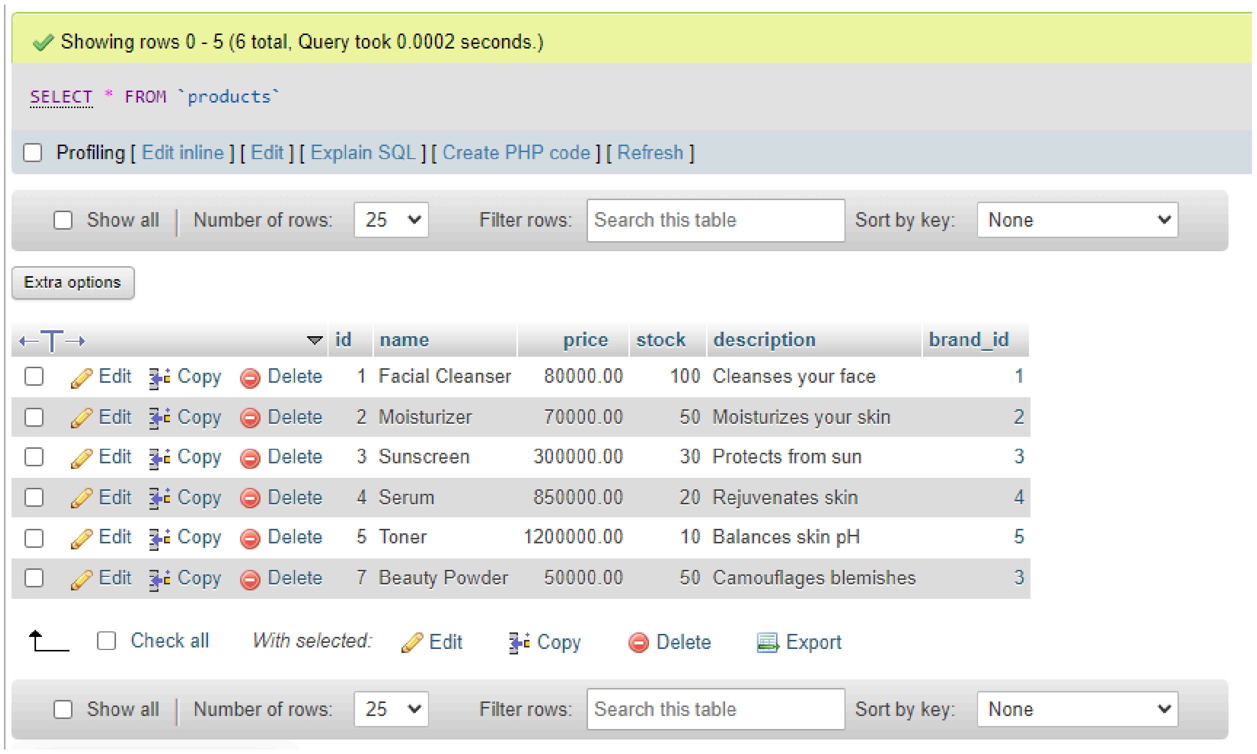
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

id	order_id	destination	status	time_issued	time_completed
1	7	Jalan Kuda 7	Delivered	2024-07-20 00:07:24	2024-07-20 00:36:55

.

- DELETE

SEBELUM


✓ Showing rows 0 - 5 (6 total, Query took 0.0002 seconds.)

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

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

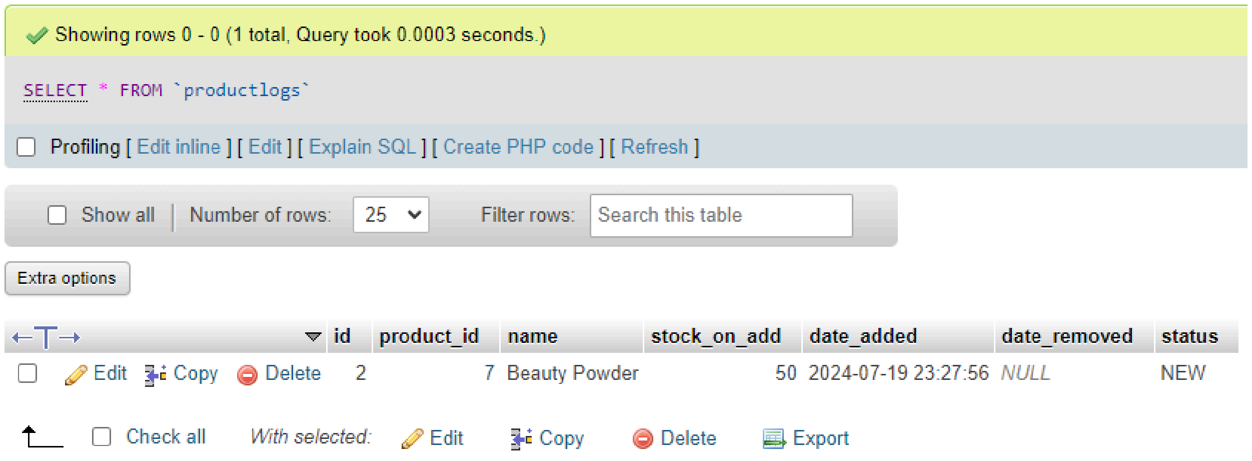
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	id	name	price	stock	description	brand_id
Edit Copy Delete	1	Facial Cleanser	80000.00	100	Cleanses your face	1
Edit Copy Delete	2	Moisturizer	70000.00	50	Moisturizes your skin	2
Edit Copy Delete	3	Sunscreen	300000.00	30	Protects from sun	3
Edit Copy Delete	4	Serum	850000.00	20	Rejuvenates skin	4
Edit Copy Delete	5	Toner	1200000.00	10	Balances skin pH	5
Edit Copy Delete	7	Beauty Powder	50000.00	50	Camouflages blemishes	3

Check all With selected: Edit Copy Delete Export

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


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

```
SELECT * FROM `productlogs`
```

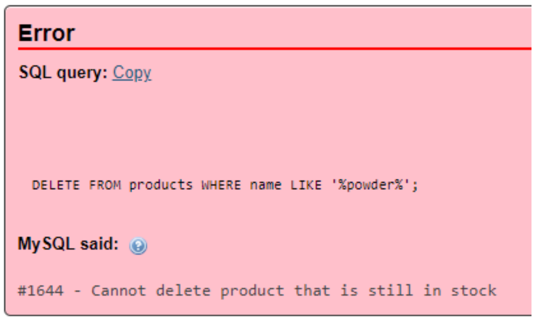
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

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

Extra options

	id	product_id	name	stock_on_add	date_added	date_removed	status
Edit Copy Delete	2	7	Beauty Powder	50	2024-07-19 23:27:56	NULL	NEW

Check all With selected: Edit Copy Delete Export


Error

SQL query: Copy

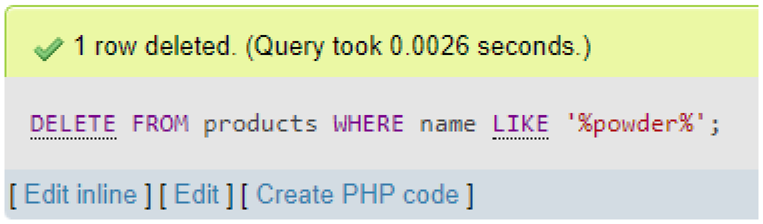
```
DELETE FROM products WHERE name LIKE '%powder%';
```

MySQL said:

```
#1644 - Cannot delete product that is still in stock
```

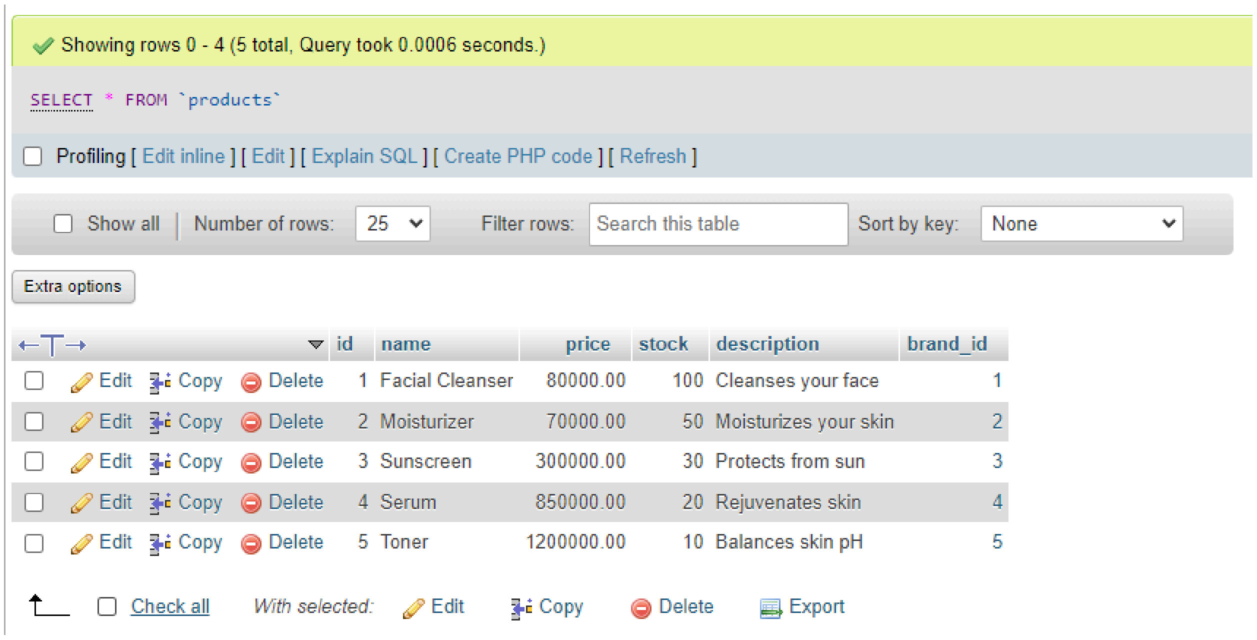
SETELAH DELETE

this is after changing the stock count of Beauty Powder to 0.



1 row deleted. (Query took 0.0026 seconds.)

```
DELETE FROM products WHERE name LIKE '%powder%';
```



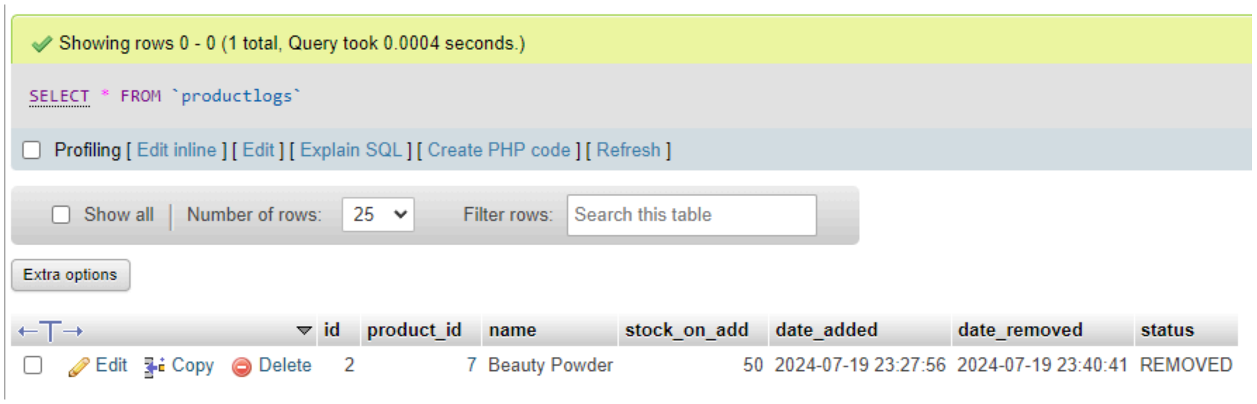
Showing rows 0 - 4 (5 total, Query took 0.0006 seconds.)

```
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

	id	name	price	stock	description	brand_id
Edit Copy Delete	1	Facial Cleanser	80000.00	100	Cleanses your face	1
Edit Copy Delete	2	Moisturizer	70000.00	50	Moisturizes your skin	2
Edit Copy Delete	3	Sunscreen	300000.00	30	Protects from sun	3
Edit Copy Delete	4	Serum	850000.00	20	Rejuvenates skin	4
Edit Copy Delete	5	Toner	1200000.00	10	Balances skin pH	5

Check all With selected: Edit Copy Delete Export



Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

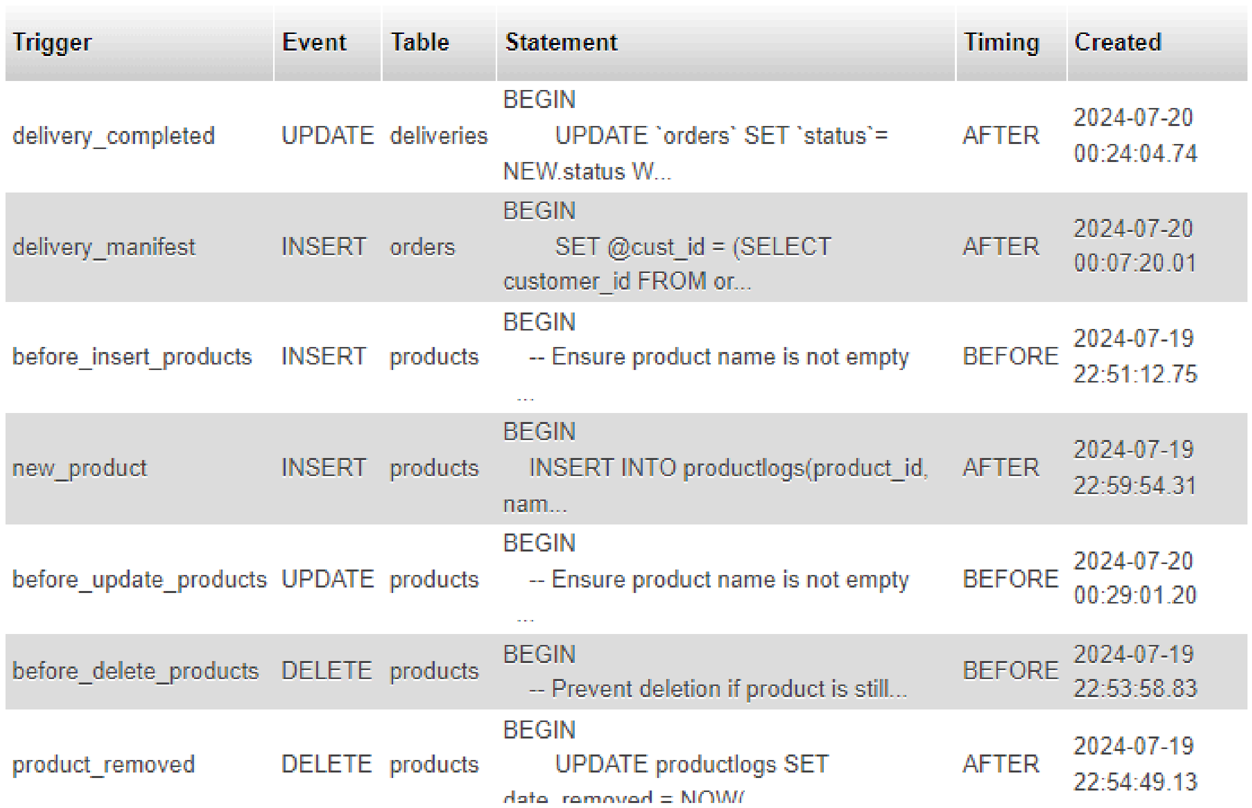
```
SELECT * FROM `productlogs`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

	id	product_id	name	stock_on_add	date_added	date_removed	status
Edit Copy Delete	2	7	Beauty Powder	50	2024-07-19 23:27:56	2024-07-19 23:40:41	REMOVED

.

4. Show list of all triggers



Trigger	Event	Table	Statement	Timing	Created
delivery_completed	UPDATE	deliveries	BEGIN UPDATE \`orders\` SET \`status\`= NEW.status W...	AFTER	2024-07-20 00:24:04.74
delivery_manifest	INSERT	orders	BEGIN SET @cust_id = (SELECT customer_id FROM or...	AFTER	2024-07-20 00:07:20.01
before_insert_products	INSERT	products	BEGIN -- Ensure product name is not empty ...	BEFORE	2024-07-19 22:51:12.75
new_product	INSERT	products	BEGIN INSERT INTO productlogs(product_id, nam...	AFTER	2024-07-19 22:59:54.31
before_update_products	UPDATE	products	BEGIN -- Ensure product name is not empty ...	BEFORE	2024-07-20 00:29:01.20
before_delete_products	DELETE	products	BEGIN -- Prevent deletion if product is still...	BEFORE	2024-07-19 22:53:58.83
product_removed	DELETE	products	BEGIN UPDATE productlogs SET date_removed = NOW(	AFTER	2024-07-19 22:54:49.13

.

INDEX

Final Project skincarestore.sql | Create Composite Index and display indexes

1. Create 3 indexes each by: creating a new table, creating an index with CREATE INDEX, creating an index with ALTER TABLE. In the index, use composite key (key with more than 1 Columns)

Answer:

1. Create Index by creating a new table

```
CREATE TABLE Accounts (
    account_id SERIAL PRIMARY KEY,
    customer_id INT NOT NULL,
    account_name varchar(255) NOT NULL,
    account_type varchar(50) NOT NULL,
    open_date DATE NOT NULL,
    balance DECIMAL(10, 2) NOT NULL,
    status varchar(50) NOT NULL,
    INDEX idx_customer_account (customer_id, account_type)
);
```

2. Create Index with CREATE Index

```
CREATE INDEX idx_account_name_status
ON accounts (account_name ASC, status);
```

3. Creating an Index with ALTER TABLE

```
ALTER TABLE accounts
ADD INDEX idx_account_balance (account_name, balance );
```

4. Show the Index list

```
SHOW INDEXES FROM accounts;
```

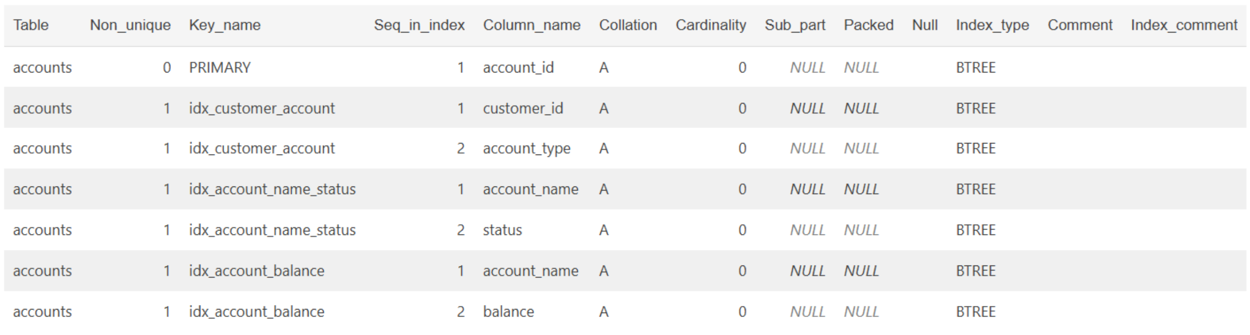


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
accounts	0	PRIMARY	1	account_id	A	0	NULL	NULL		BTREE		
accounts	1	idx_customer_account	1	customer_id	A	0	NULL	NULL		BTREE		
accounts	1	idx_customer_account	2	account_type	A	0	NULL	NULL		BTREE		
accounts	1	idx_account_name_status	1	account_name	A	0	NULL	NULL		BTREE		
accounts	1	idx_account_name_status	2	status	A	0	NULL	NULL		BTREE		
accounts	1	idx_account_balance	1	account_name	A	0	NULL	NULL		BTREE		
accounts	1	idx_account_balance	2	balance	A	0	NULL	NULL		BTREE		

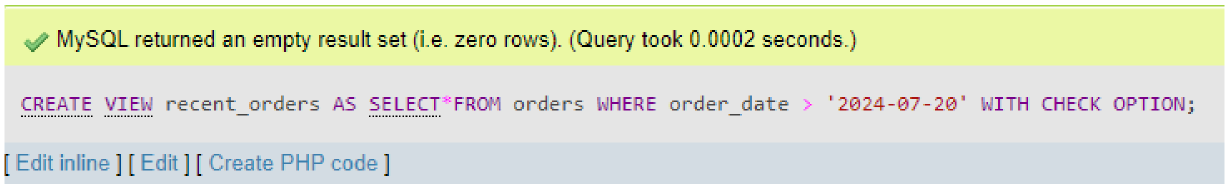
VIEW

Final Project skincarestore.sql | Create a VIEW, and perform updates

1. Create 3 views and use the WITH CHECK OPTION clause either cascaded or local

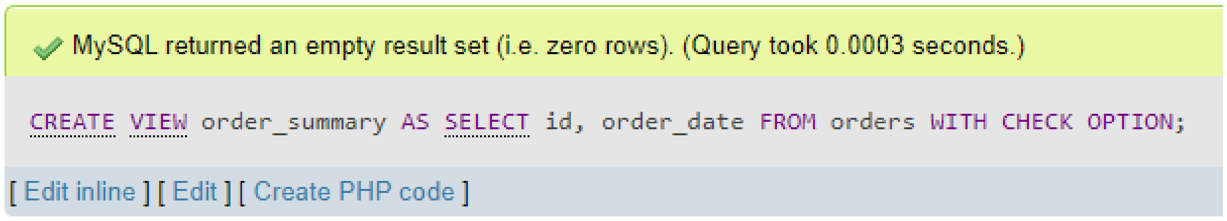
Answer:

```
CREATE VIEW recent_orders AS
SELECT*FROM orders
WHERE order_date > '2024-07-20'
WITH CHECK OPTION;
```



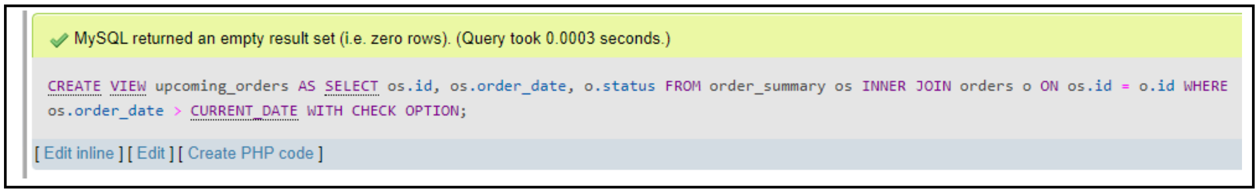
```
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

CREATE VIEW recent_orders AS SELECT*FROM orders WHERE order_date > '2024-07-20' WITH CHECK OPTION;

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

```
CREATE VIEW order_summary AS
SELECT id, order_date
FROM orders
WITH CHECK OPTION;
```



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

CREATE VIEW order_summary AS SELECT id, order_date FROM orders WITH CHECK OPTION;

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

```
CREATE VIEW upcoming_orders AS
SELECT os.id, os.order_date, o.status
FROM order_summary os
INNER JOIN orders o ON os.id = o.id
WHERE os.order_date > CURRENT_DATE
WITH CHECK OPTION;
```


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

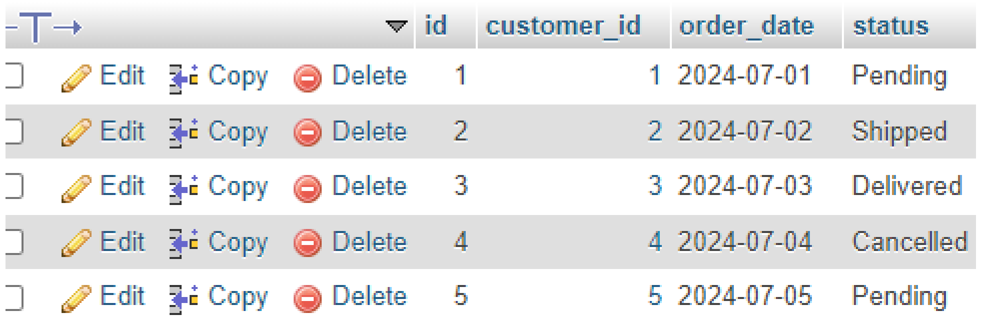
```
CREATE VIEW upcoming_orders AS SELECT os.id, os.order_date, o.status FROM order_summary os INNER JOIN orders o ON os.id = o.id WHERE
os.order_date > CURRENT_DATE WITH CHECK OPTION;
```

2. Update dan insert

Answer:

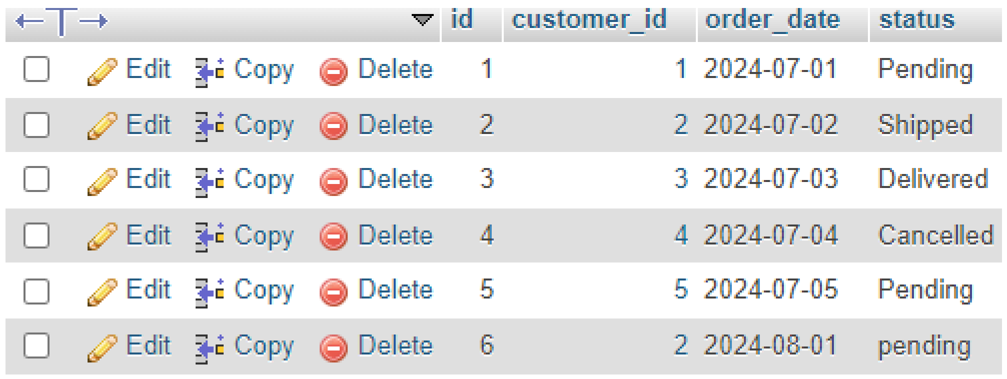
INSERT INTO recent_orders (customer_id, order_date, status)
VALUES (2, '2024-08-01', 'pending');

sebelum



id	customer_id	order_date	status
1	1	2024-07-01	Pending
2	2	2024-07-02	Shipped
3	3	2024-07-03	Delivered
4	4	2024-07-04	Cancelled
5	5	2024-07-05	Pending

sesudah

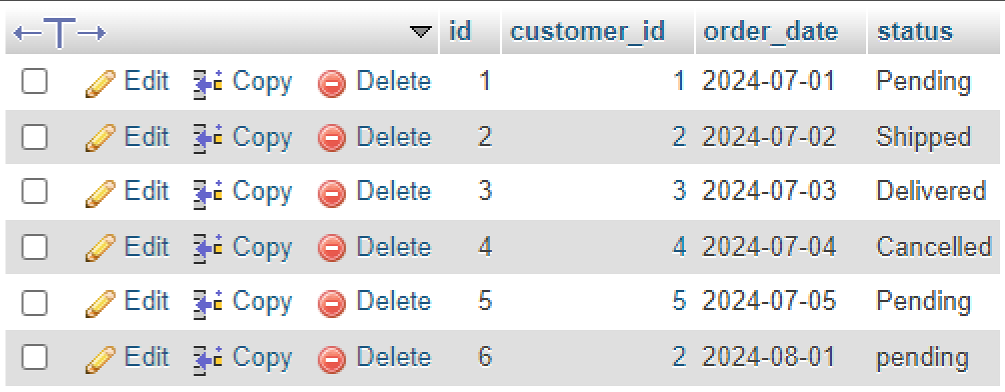


id	customer_id	order_date	status
1	1	2024-07-01	Pending
2	2	2024-07-02	Shipped
3	3	2024-07-03	Delivered
4	4	2024-07-04	Cancelled
5	5	2024-07-05	Pending
6	2	2024-08-01	pending

Answer:

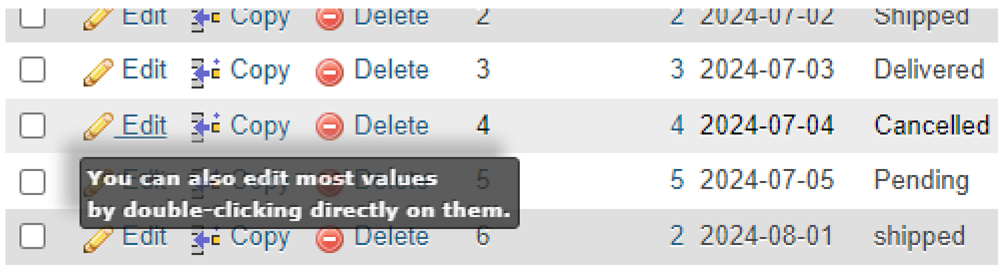
UPDATE upcoming_orders
SET status = 'shipped'
WHERE id = 6;

BEFORE



				id	customer_id	order_date	status
☐	Edit	Copy	Delete	1	1	2024-07-01	Pending
☐	Edit	Copy	Delete	2	2	2024-07-02	Shipped
☐	Edit	Copy	Delete	3	3	2024-07-03	Delivered
☐	Edit	Copy	Delete	4	4	2024-07-04	Cancelled
☐	Edit	Copy	Delete	5	5	2024-07-05	Pending
☐	Edit	Copy	Delete	6	2	2024-08-01	pending

AFTER



☐	Edit	Copy	Delete	2	2	2024-07-02	Shipped
☐	Edit	Copy	Delete	3	3	2024-07-03	Delivered
☐	Edit	Copy	Delete	4	4	2024-07-04	Cancelled
☐				5	5	2024-07-05	Pending
☐	Edit	Copy	Delete	6	2	2024-08-01	shipped

You can also edit most values
by double-clicking directly on them.

3. Show list view

Answer:

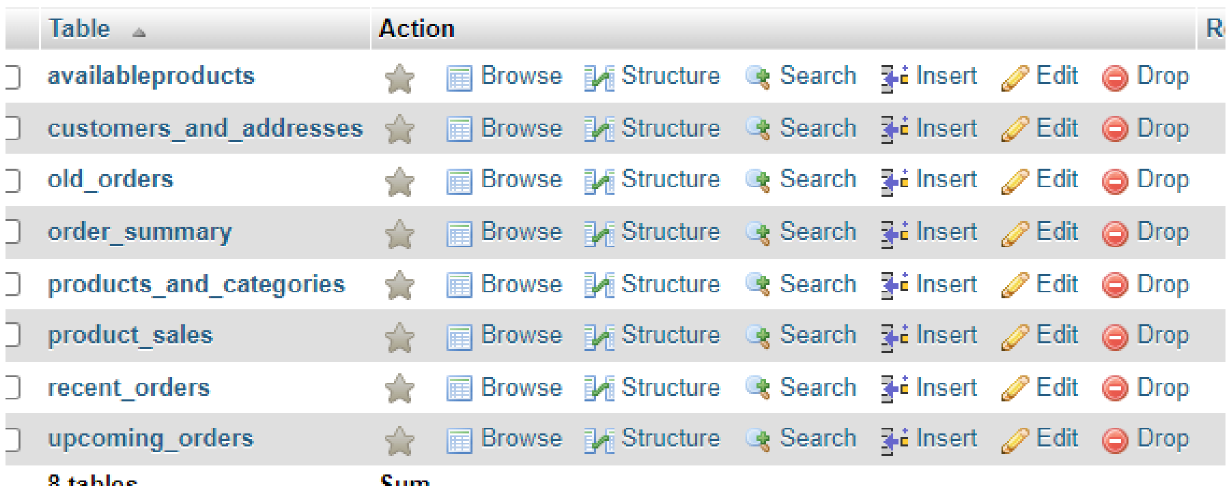


	Table	Action							R
☐	availableproducts	☆	Browse	Structure	Search	Insert	Edit	Drop	
☐	customers_and_addresses	☆	Browse	Structure	Search	Insert	Edit	Drop	
☐	old_orders	☆	Browse	Structure	Search	Insert	Edit	Drop	
☐	order_summary	☆	Browse	Structure	Search	Insert	Edit	Drop	
☐	products_and_categories	☆	Browse	Structure	Search	Insert	Edit	Drop	
☐	product_sales	☆	Browse	Structure	Search	Insert	Edit	Drop	
☐	recent_orders	☆	Browse	Structure	Search	Insert	Edit	Drop	
☐	upcoming_orders	☆	Browse	Structure	Search	Insert	Edit	Drop	
	8 tables	Sum							

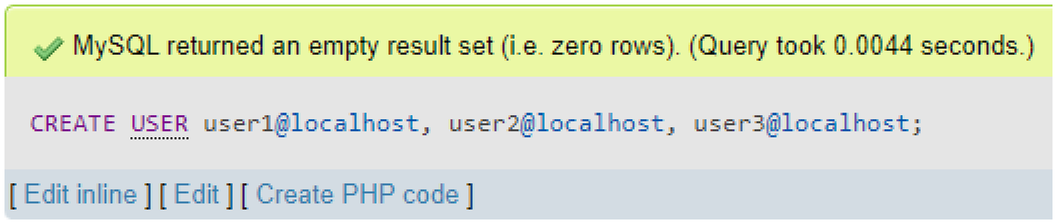
DATABASE SECURITY

Final Project RentalMobil.sql | Create USER-ROLE- and give Privilege.

1. Create 3 new users with the names user1, user2, and user3

Answer:

```
CREATE USER user1@localhost, user2@localhost, user3@localhost
```


MySQL returned an empty result set (i.e. zero rows). (Query took 0.0044 seconds.)

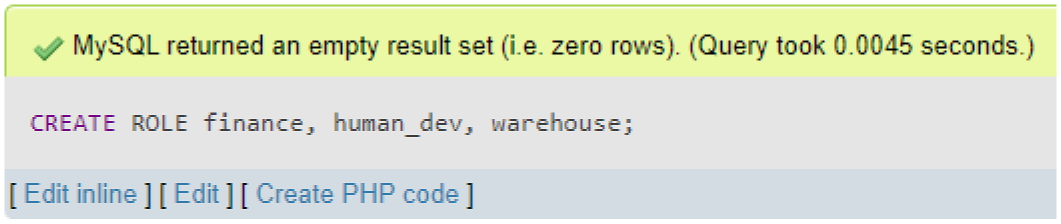
```
CREATE USER user1@localhost, user2@localhost, user3@localhost;
```

create 3 new users with the names user1, user2, and user3 by using the CREATE USER command.

2. Create 3 new roles finance, human_dev, warehouse

Answer:

```
CREATE ROLE finance, human_dev, warehouse;
```

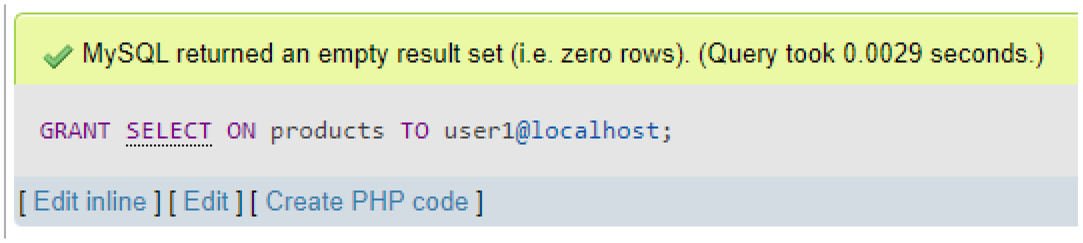

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0045 seconds.)

```
CREATE ROLE finance, human_dev, warehouse;
```

3. Add privileges to user1 so that he can access one of the tables in the database.

Answer:

```
GRANT SELECT
ON products
TO user1@localhost
```


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

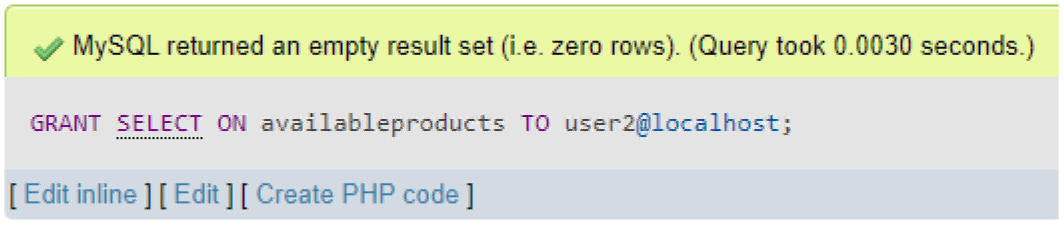
```
GRANT SELECT ON products TO user1@localhost;
```

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

4. Add privileges to user2 to access one of the views in your database

Answer:

```
GRANT SELECT
ON availableproducts
TO user2@localhost
```


MySQL returned an empty result set (i.e. zero rows). (Query took 0.0030 seconds.)

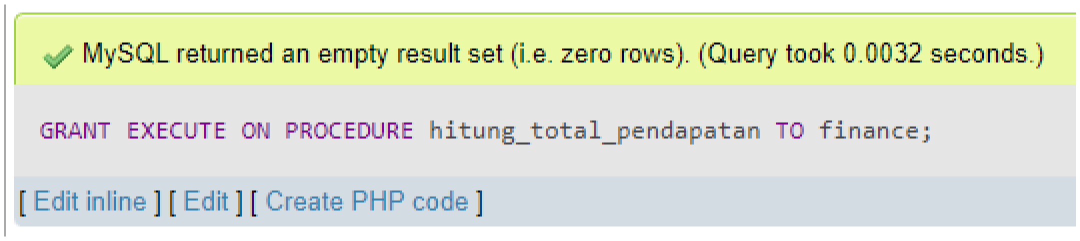
```
GRANT SELECT ON availableproducts TO user2@localhost;
```

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

5. Add privileges to finance to access one of the procedures in your database

Answer:

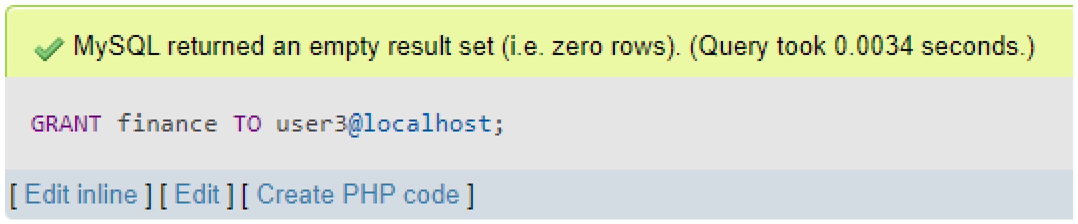
```
GRANT EXECUTE
ON PROCEDURE hitung_total_pendapatan
TO finance
GRANT finance
TO user3@localhost
```



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0032 seconds.)

```
GRANT EXECUTE ON PROCEDURE hitung_total_pendapatan TO finance;
```

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



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0034 seconds.)

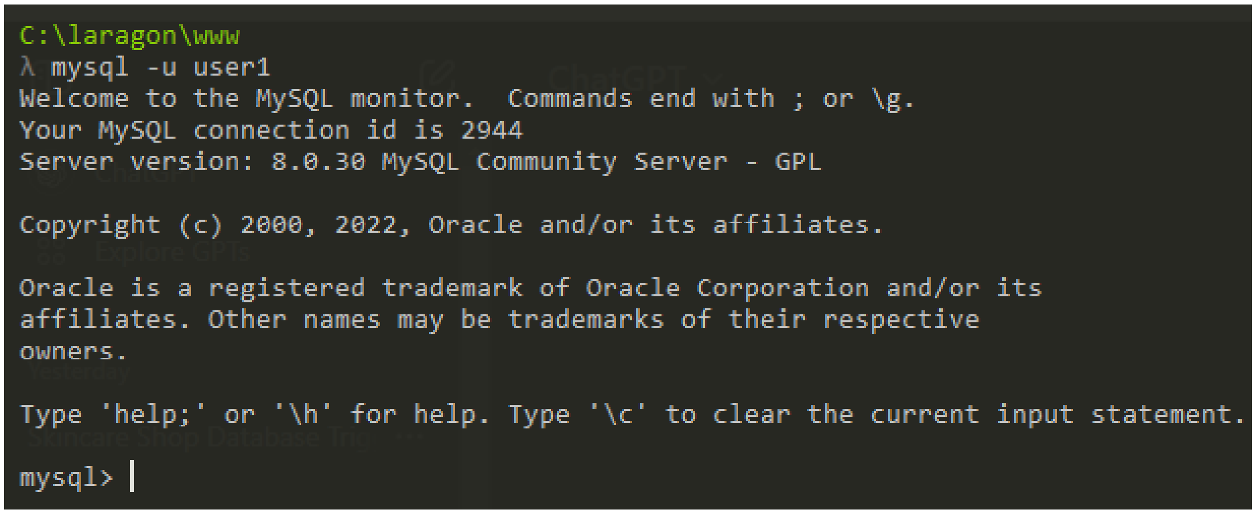
```
GRANT finance TO user3@localhost;
```

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

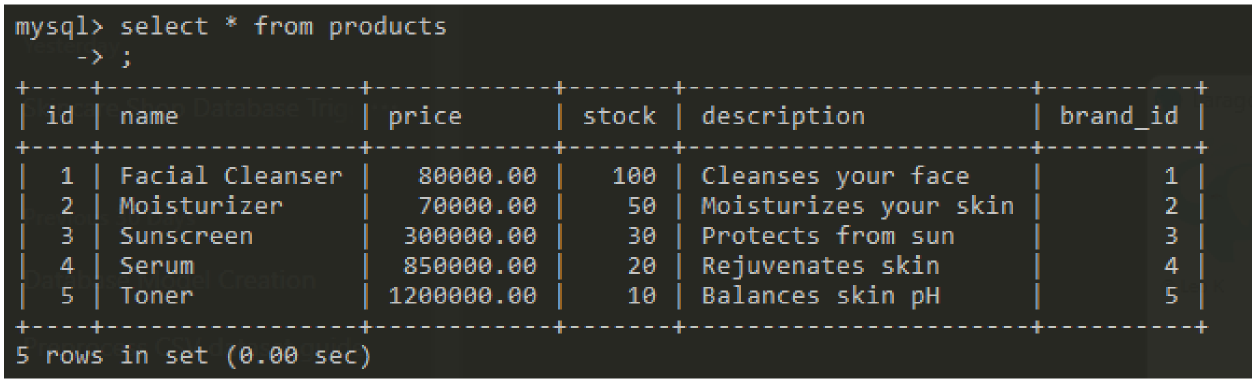
6. Execute to prove that the privileges have been successfully assigned to the user and role

   a. Login as user1

```
mysql -u user1

select * from products
```



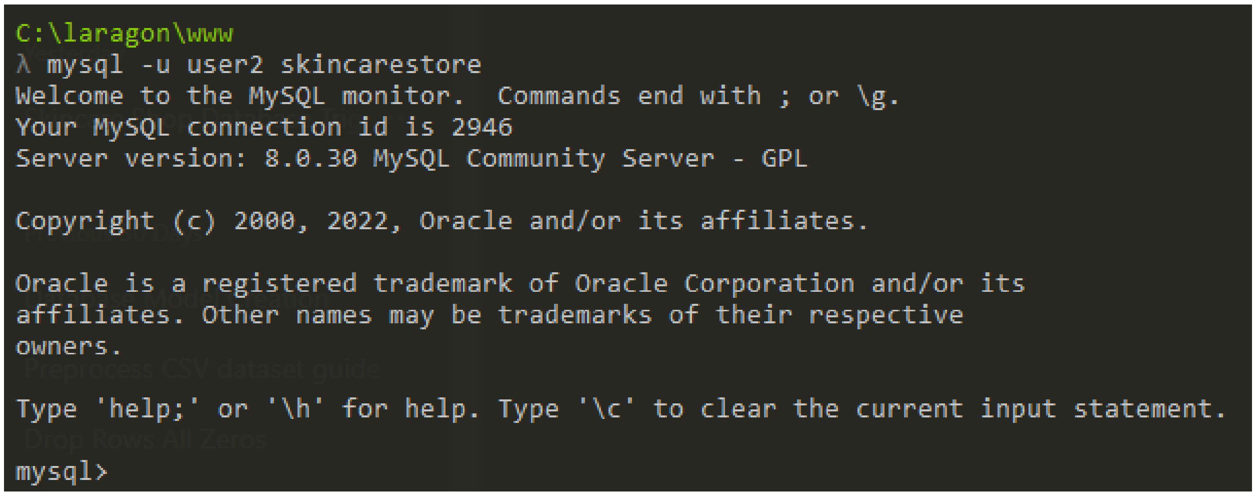
```
C:\laragon\www
λ mysql -u user1
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2944
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```



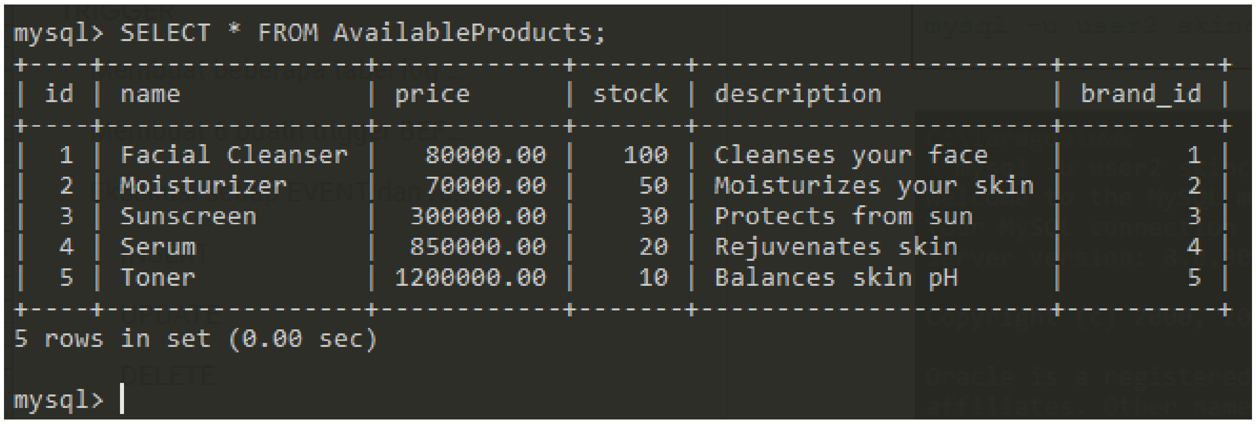
```
mysql> select * from products
    -> ;
+----+-----------------+------------+-------+-------------------------+----------+
| id | name            | price      | stock | description             | brand_id |
+----+-----------------+------------+-------+-------------------------+----------+
|  1 | Facial Cleanser |   80000.00 |   100 | Cleanses your face      |        1 |
|  2 | Moisturizer     |   70000.00 |    50 | Moisturizes your skin   |        2 |
|  3 | Sunscreen       |  300000.00 |    30 | Protects from sun       |        3 |
|  4 | Serum           |  850000.00 |    20 | Rejuvenates skin        |        4 |
|  5 | Toner           | 1200000.00 |    10 | Balances skin pH        |        5 |
+----+-----------------+------------+-------+-------------------------+----------+
5 rows in set (0.00 sec)
```

b. Login as user2

```
mysql -u user2 skincarestore

SELECT * FROM AvailableProducts;
```



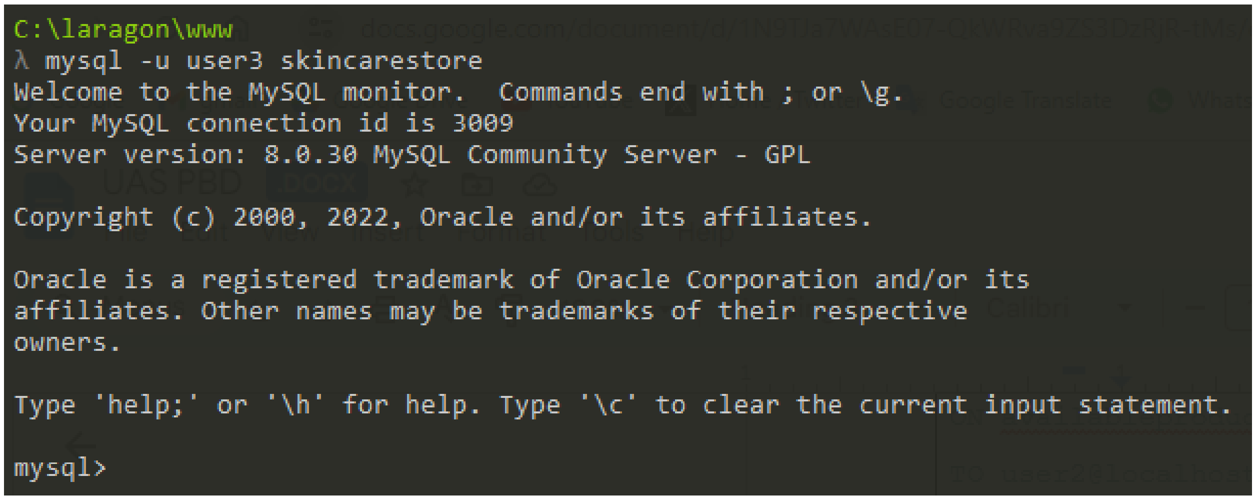
```
C:\laragon\www
λ mysql -u user2 skincarestore
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2946
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```



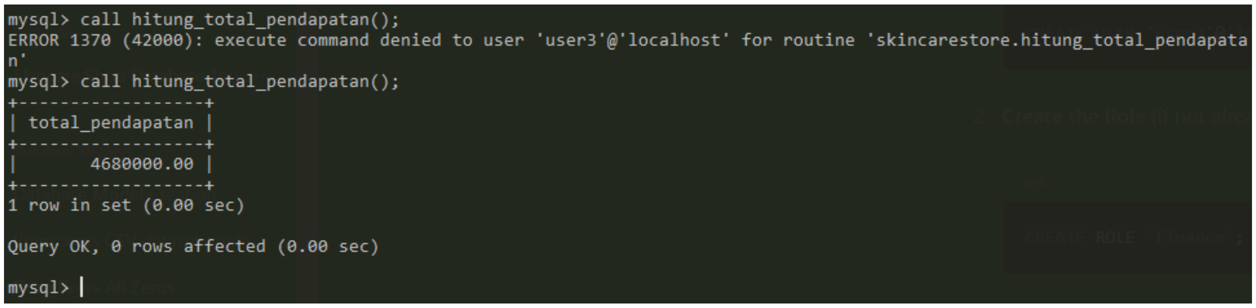
```
mysql> SELECT * FROM AvailableProducts;
+----+-----------------+------------+-------+-------------------------+----------+
| id | name            | price      | stock | description             | brand_id |
+----+-----------------+------------+-------+-------------------------+----------+
|  1 | Facial Cleanser |   80000.00 |   100 | Cleanses your face      |        1 |
|  2 | Moisturizer     |   70000.00 |    50 | Moisturizes your skin   |        2 |
|  3 | Sunscreen       |  300000.00 |    30 | Protects from sun       |        3 |
|  4 | Serum           |  850000.00 |    20 | Rejuvenates skin        |        4 |
|  5 | Toner           | 1200000.00 |    10 | Balances skin pH        |        5 |
+----+-----------------+------------+-------+-------------------------+----------+
5 rows in set (0.00 sec)

mysql>
```

c. Login as user3

```
mysql -u user3 skincarestore
CALL hitung_total_pendapatan();
```



```
C:\laragon\www
λ mysql -u user3 skincarestore
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3009
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```



```
mysql> call hitung_total_pendapatan();
ERROR 1370 (42000): execute command denied to user 'user3'@'localhost' for routine 'skincarestore.hitung_total_pendapata
n'
mysql> call hitung_total_pendapatan();
+-------------------+
| total_pendapatan |
+-------------------+
|       4680000.00 |
+-------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```