Explain and analysis (if necessary) all the query of project:

--admin\_login

select \* from admin\_table where admin\_name = $user\_name

This query is used to check the data that we input existed in the admin table. If yes then it is a valid admin account. If not then it will tell us that the username or password is incorrect.

--admin\_registration

SELECT \* FROM admin\_table WHERE admin\_email = '$email';

INSERT INTO admin\_table (admin\_name, admin\_email, admin\_password) VALUES ('$username','$email','$password')

This query is used to check the data (email) that we input existed in the admin table. If yes, then it will tell us that the user already existed. If no then we insert the data.

--delete\_brand

DELETE FROM brands WHERE brand\_id = '$delete\_id'

This query is used to delete brand from the table.

--delete\_category

DELETE FROM categories WHERE category\_id = '$delete\_id'

This query is used to delete category from the table.

--delete\_order

DELETE FROM orders WHERE order\_id = '$delete\_id'

This query is used to delete order from the table.

--delete\_payment

DELETE FROM payments WHERE payment\_id = '$delete\_id'

This query is used to delete payment from the table.

--delete\_product

DELETE FROM products WHERE product\_id = '$delete\_id'

This query is used to delete product from the table.

--delete\_user

DELETE FROM user\_table WHERE x = '$delete\_id'

This query is used to delete user from the table.

--edit\_brand

SELECT \* FROM brands WHERE brand\_id = '$brand\_id'

UPDATE brands SET brand\_title = '$brand\_title' WHERE brand\_id = '$brand\_id'

This query check the existence current editing brand. By fetching that, it set the new value equal the old value (we do that to ensure that after update, the untouched part remain the same). After that, it update the new value.

--edit\_category

SELECT \* FROM categories WHERE category\_id = '$category\_id'

UPDATE categories SET category\_title = '$category\_title' WHERE category\_id = '$category\_id'

This query check the existence current editing category. By fetching that, it set the new value equal the old value (we do that to ensure that after update, the untouched part remain the same). After that, it update the new value.

--edit\_product

SELECT \* FROM products WHERE product\_id = $edit\_id

SELECT \* FROM categories WHERE category\_id = $product\_category

SELECT \* FROM brands WHERE brand\_id = $product\_brand

UPDATE products SET product\_title = '$product\_title',

                    product\_description = '$product\_description', product\_keywords = '$product\_keywords',

                    category\_id = '$product\_category', brand\_id = '$product\_brand', product\_image1 = '$product\_image1',

                    product\_image2 = '$product\_image2', product\_image3 = '$product\_image3', product\_price = '$product\_price'

                    WHERE product\_id = $edit\_id;

This query check the existence of current editing product. By fetching that, it set the new value equal the old value (we do that to ensure that after update, the untouched part remain the same). After that, it update the new value.

--insert\_brand

SELECT \* FROM brands WHERE brand\_title = '$brand\_title'

INSERT INTO brands (brand\_title) VALUES ('$brand\_title')

This query check the existence of the going to insert brand. If it is already existed, then do not insert. If not, then insert the new record.

--insert\_categories

SELECT \* FROM categories WHERE category\_title = '$category\_title'

INSERT INTO categories (category\_title) VALUES ('$category\_title')

This query check the existence of the going to insert category. If it is already existed, then do not insert. If not, then insert the new record.

--insert\_product

INSERT INTO products (product\_title, product\_description, product\_keywords, category\_id, brand\_id, product\_image1, product\_image2, product\_image3, product\_price, date, status)

            VALUES ('$product\_title','$product\_description','$product\_keywords','$product\_categories','$product\_brands','$product\_image1','$product\_image2','$product\_image3','$product\_price',NOW(), '$product\_status')

This query insert the new record from backend to the database.

--list\_orders

SELECT \* FROM user\_orders JOIN user\_table ON user\_orders.user\_id = user\_table.user\_id  ORDER BY order\_id DESC

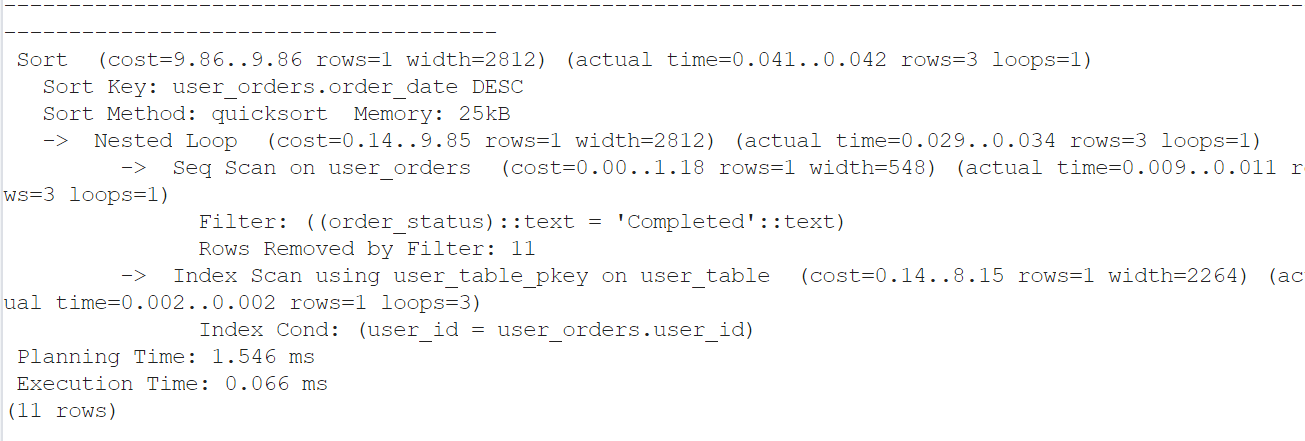
 WHERE order\_status = '$order\_status\_filter'

AND order\_date BETWEEN '$start\_date' AND '$end\_date';

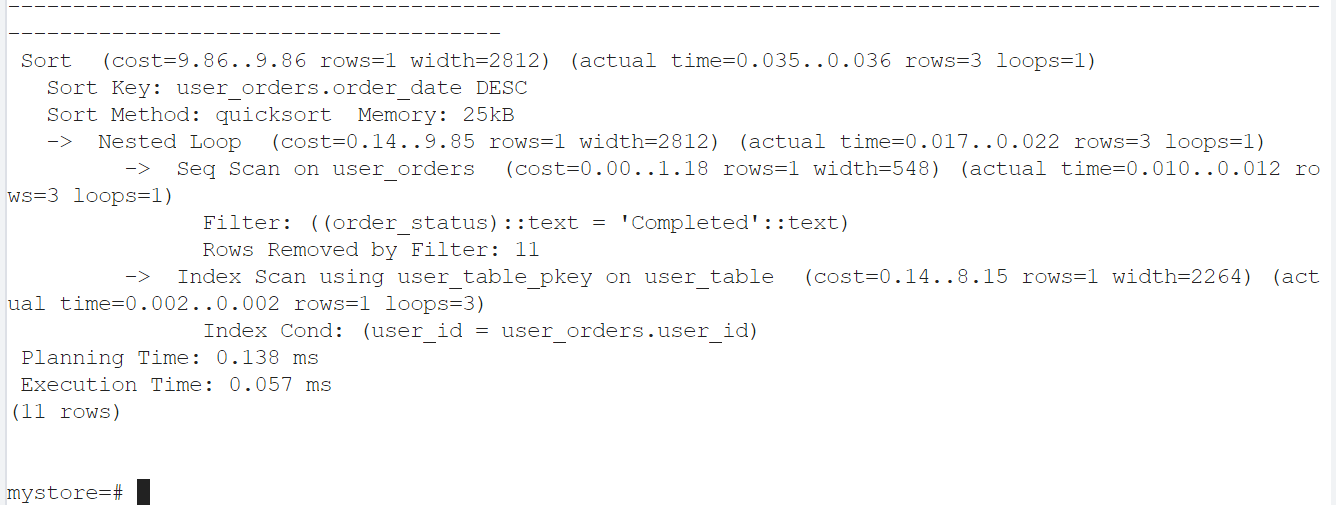
This query is used to fetch the data from two table user\_table and user\_order. The web will present it after filter by date and status.

Analysis:

With index in order\_date:



Without index in order\_date.



We can see that the speed have change a litter by adding index. The planning increase because the index btree did not increase the performance of the statement (btree only help improve the sorting and searching the left most column of index) .

--list\_payment

SELECT \* FROM user\_payments JOIN user\_orders ON user\_payments.order\_id = user\_orders.order\_id

WHERE payment\_mode = '$payment\_method\_filter'

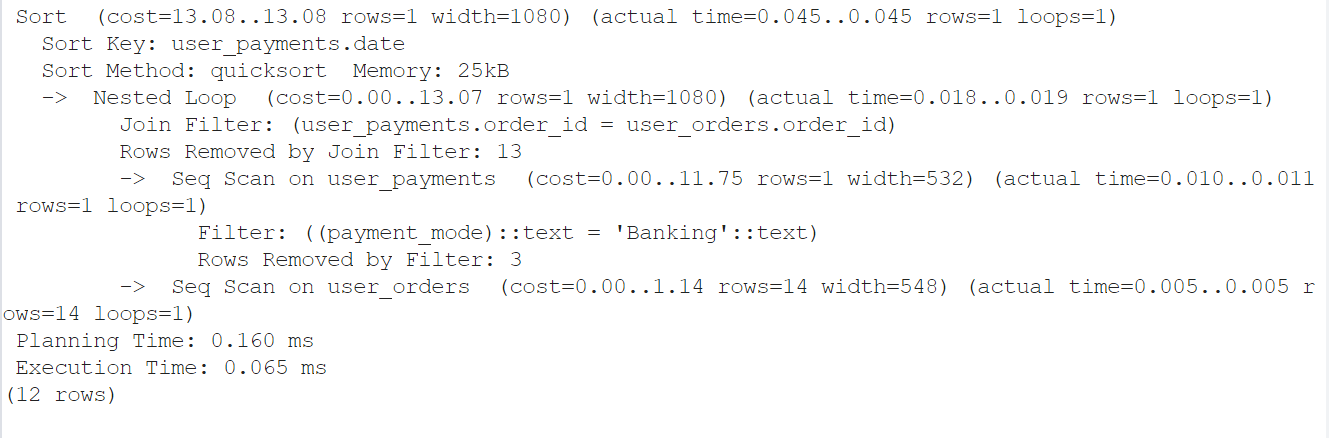
AND date BETWEEN '$start\_date' AND '$end\_date';

SELECT get\_total\_sales('$start\_date', '$end\_date', '$payment\_mode\_filter') AS total\_sales;

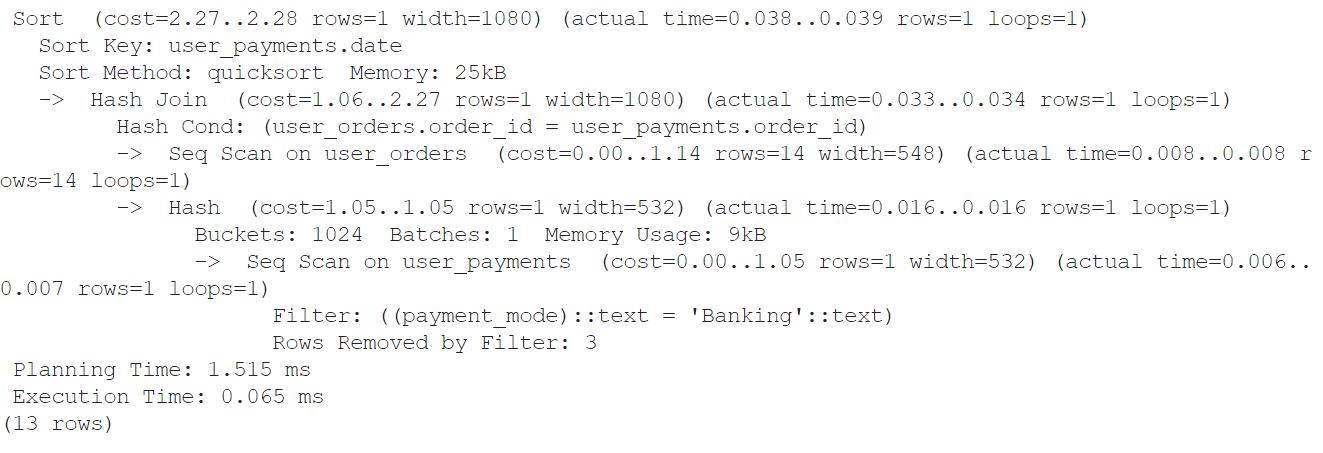
This query is used to fetch the data from two table user\_payment and user\_order. The web will present it after filter by date and status.

Analysis:

Without index in payment date:



With index:



We can see that the speed have change a litter by adding index. The planning increase because the index btree did not increase the performance of the statement (btree only help improve the sorting and searching the left most column of index)

--list\_user

SELECT \* FROM user\_table;

This query is used to fetch data from table user\_table.

--view\_brand

SELECT \* FROM brands;

This query is used to fetch data from table brands.

--view\_categories

SELECT \* FROM categories;

This query is used to fetch data from table categories.

--view\_products

SELECT \* FROM products;

This query is used to fetch data from table products.

----Function

--get\_product

SELECT \* FROM products ORDER BY RANDOM() LIMIT 9 OFFSET 0;

This query is used to fetch 9 random record from table products.

--get\_all\_product

SELECT \* FROM products ORDER BY RANDOM();

This query is used to fetch all the record from table products.

--filter

SELECT \* FROM brands;

SELECT \* FROM categories;

SELECT \* FROM products WHERE 1=1

AND category\_id IN ('" . implode("','", $category\_ids) . "')

AND brand\_id::int IN ('" . implode("','", $brand\_ids) . "')

AND CAST(product\_price AS FLOAT) >=  (float)$min\_price

AND CAST(product\_price AS FLOAT) <=  (float)$max\_price

ORDER BY

    CAST(product\_price AS FLOAT) ASC;

    CAST(product\_price AS FLOAT) DESC;

    product\_title ASC;

    product\_title DESC;

    amount\_sold DESC;

    amount\_sold ASC;

First, we take the brand and categories to build a check box. After that, we start the filter. At first, if there are no brand, category or any other option selected, we take all the record. But if there are any selected brand or category, we add the condition to the query and fetching. Same with the max and min price. The order of presentation product is added if there is any input in sorting condition.

--search

SELECT \* FROM products WHERE product\_keywords LIKE '%$search\_data\_value%'

The query is used to fetch any record in product that have matching keyword include search\_data\_value.

--view\_detail

SELECT \* FROM products WHERE product\_id=$product\_id

We fetch all the information of the specific product.

--cart\_function

SELECT \* FROM cart\_details WHERE ip\_address='$ip' AND product\_id='$get\_product\_id';

INSERT INTO cart\_details (product\_id,ip\_address) VALUES ('$get\_product\_id','$ip')

We first check if the product already added to users card. If yes, then we refuse add the product to cart table. If yes then we insert the record.

SELECT CAST(SUM(p.product\_price \* c.quantity) AS NUMERIC(10, 2)) AS total\_price

    FROM cart\_details c

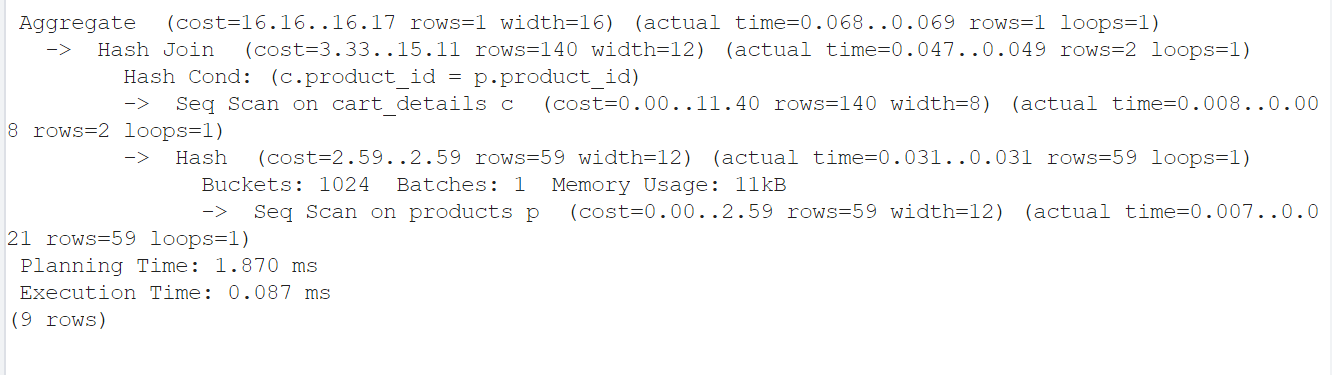
    JOIN products p ON c.product\_id = p.product\_id

    WHERE c.ip\_address = '$get\_ip\_add';

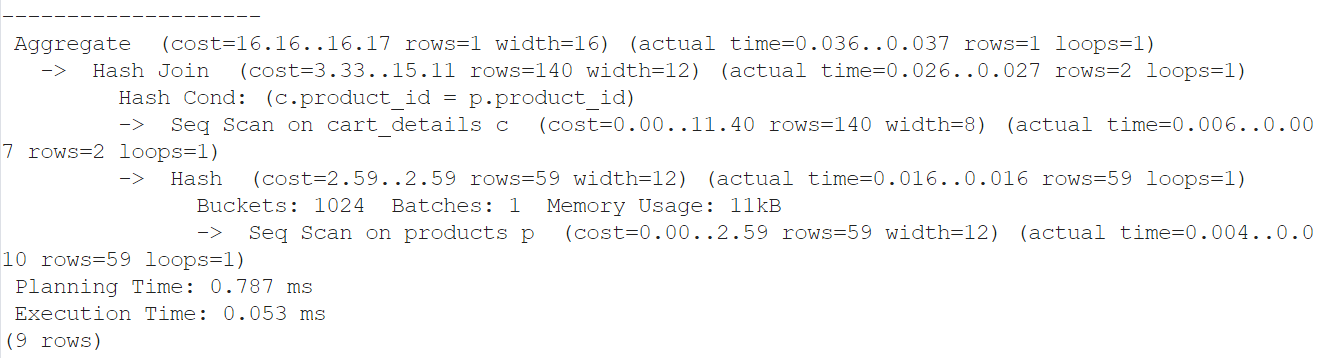
We calculate the price of all item in the cart by select all the product in current (check condition by ip) then multiple the quantity save in cart and the price save in product.

Analysis:

With index in product\_id and user\_id:



Without index:



The time need to executed and planning reduce because the index in here improve the performance in searching that use the product\_id (btree only help improve the sorting and searching the left most column of index).

----------

--confirm\_payment

SELECT \* FROM user\_orders WHERE order\_id = '$order\_id';

INSERT INTO user\_payments (order\_id, payment\_mode)

                    VALUES ('$order\_id', '$payment\_mode');

UPDATE user\_orders SET order\_status = '$complete' WHERE order\_id = '$order\_id'

We check the current order existence and set all current value to old value. The inpuy in payment will insert a new record in the payment table. After that, we update the status of the order.

--delete\_account

DELETE FROM user\_table WHERE username = '$username\_session'

This query is used to delete the user account by user them self.

--edit\_account

SELECT \* FROM user\_table WHERE username = '$user\_session\_name';

UPDATE user\_table SET username='$username', user\_email='$user\_email',

                        user\_image='$user\_image', user\_address='$user\_address', user\_mobile='$user\_mobile'

                        WHERE user\_id=$update\_id;

We first fetch the record of current user from the database. After that, we set all current value to old value, and then we update the current value to the new value. We update the record.

--order

SELECT \* FROM cart\_details WHERE ip\_address = '$get\_ip\_add';

SELECT \* FROM products WHERE product\_id = $product\_id;

INSERT INTO user\_orders (user\_id, amount\_due, invoice\_number, total\_products, order\_date, order\_status) VALUES ($user\_id, $total\_price, $invoice\_number, " . count($order\_info\_values) . ", '$order\_date', '$status');

DELETE FROM cart\_details WHERE ip\_address = '$get\_ip\_add';

Select all the record that match the ip in the cart\_detail. With every product, loop through that and calculate the total price,… then insert the record in to the order\_table.

When input in delete, remove that record from cart table.

--profile

SELECT \* FROM user\_table WHERE username = '$username';

Get the record of current user from the user\_table and show all data. And call the order function and payment function.

--login

SELECT \* FROM user\_table WHERE username='$user\_username';

SELECT \* FROM cart\_details WHERE ip\_address='$user\_ip';

This query is used to check the data that we input existed in the user table. If yes then it is a valid user account. If not then it will tell us that the username or password is incorrect.

--user\_register

SELECT \* FROM user\_table WHERE username='$user\_username' OR user\_email='$user\_email';

INSERT INTO user\_table (username,user\_email,user\_password, user\_image, user\_ip, user\_address, user\_mobile)

    VALUES ('$user\_username','$user\_email','$hash\_password','$user\_image','$user\_ip','$user\_address','$user\_contact');

This query is used to check the data (email) that we input existed in the users table. If yes, then it will tell us that the user already existed. If no then we insert the data.

--user\_order

SELECT \* FROM user\_table WHERE username = '$username';

SELECT \* FROM user\_orders WHERE user\_id = $user\_id;

We use this to fetch all the data about order the user have by matching it with user\_id of each record.