

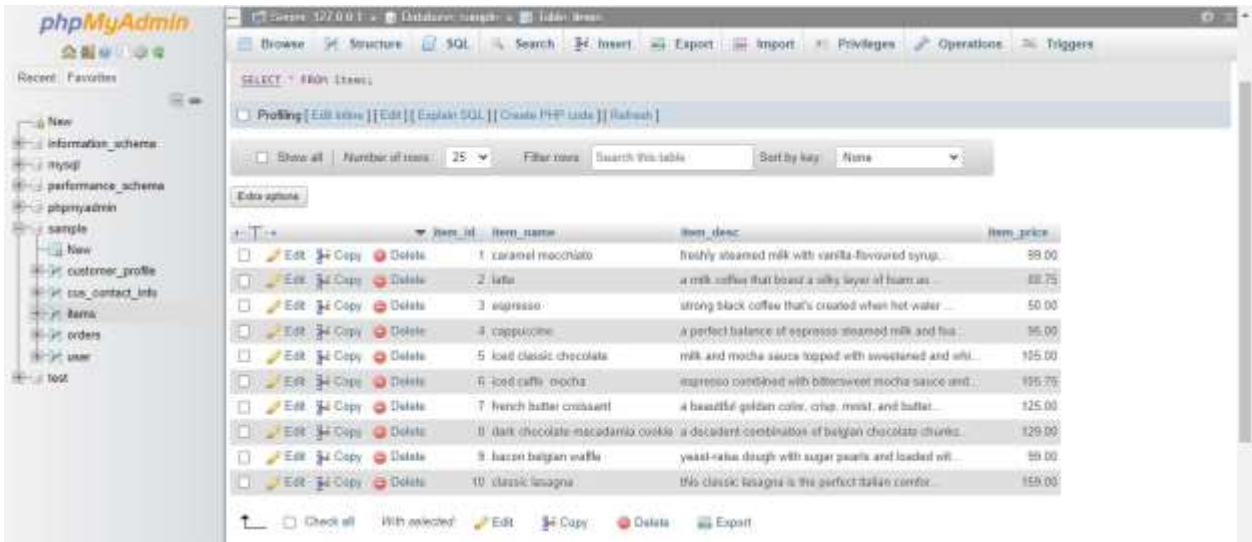


# TINY CODERS (WEB SYSTEMS)

## MEMBERS:

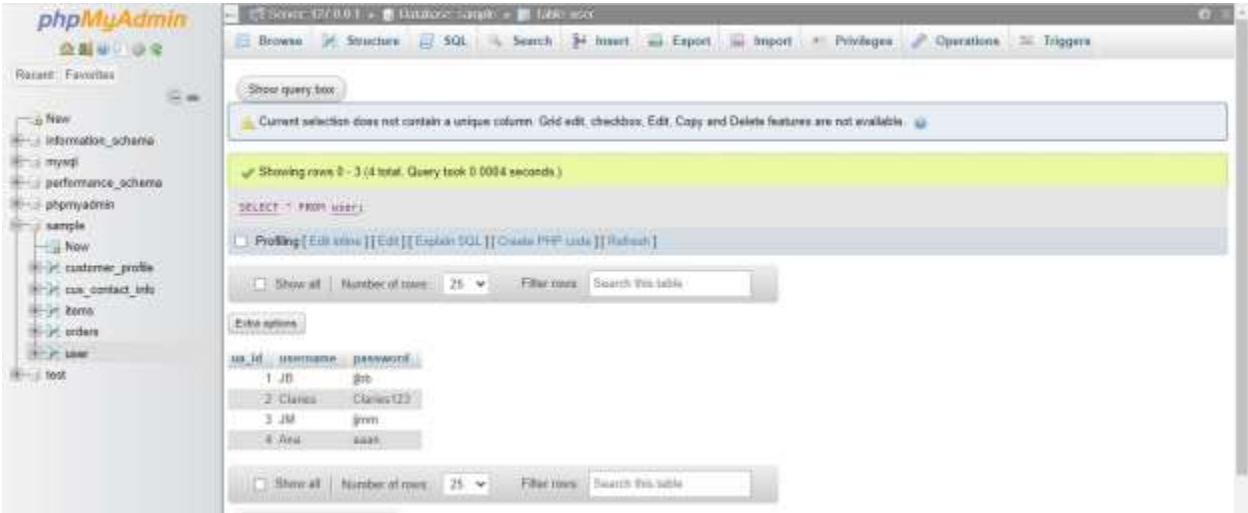
- AMISTOSO, CLARIES ANN B.
- BATA, MARIZ S.
- FETIL, JAENNIE CYRELL N.
- RESENTES, SHANLEY R.
- SABLAYAN, PENELOPE R.

## 1. DISPLAY ALL ITEMS:



Here, we inserted the item name, description as well as the item price and we display the information using the SQL “SELECT \* FROM items.”

## 2. DISPLAY ALL THE USERS:



Same with displaying the items we also used “SELECT” to display all the users in our database.



3. USING WHERE CLAUSE

The screenshot shows the phpMyAdmin interface with the 'customer\_profile' table selected. A SQL query is entered in the query box: `SELECT * FROM customer_profile WHERE gender = 'M';`. The results are displayed in a table with columns: `cus_id`, `name`, `gender`, and `iss_id`. Two rows are shown, both with gender 'M'.

| cus_id | name | gender | iss_id |
|--------|------|--------|--------|
| 1      | JB   | M      | 1      |
| 3      | JM   | M      | 3      |

Here, we view the customer\_profile table and run a SQL query to display all customers whose gender is “M”. The results show two rows, meaning there are two customers in the table with a gender of “M”.

4. DISPLAY ALL ORDERS OF CUS\_ID 1

The screenshot shows the phpMyAdmin interface with the 'orders' table selected. A SQL query is entered in the query box: `SELECT * FROM 'orders' WHERE cus_id = '1';`. The results are displayed in a table with columns: `order_id`, `item_id`, `item_qty`, and `cus_id`. Two rows are shown, both with cus\_id '1'.

| order_id | item_id | item_qty | cus_id |
|----------|---------|----------|--------|
| 1        | 4       | 2        | 1      |
| 3        | 9       | 2        | 1      |

Here, we're examining the "orders" table and ran a SQL query to retrieve all orders associated with customer ID 1. The results display two rows, confirming that this customer has placed two orders.



5. DISPLAYING TOTAL QUANTITY SOLD PER ITEMS:

| Item name               | total_qty_sold |
|-------------------------|----------------|
| bacon belgian waffle    | 2              |
| cappuccino              | 2              |
| classic leagna          | 1              |
| french butter croissant | 1              |
| loaf classic chocolate  | 1              |

Here, the total quantity sold for each product were presented, displaying only descriptive columns and total quantity sold. This enables us to present our information in a more concise and clearer perspective while putting emphasis on the quantities.

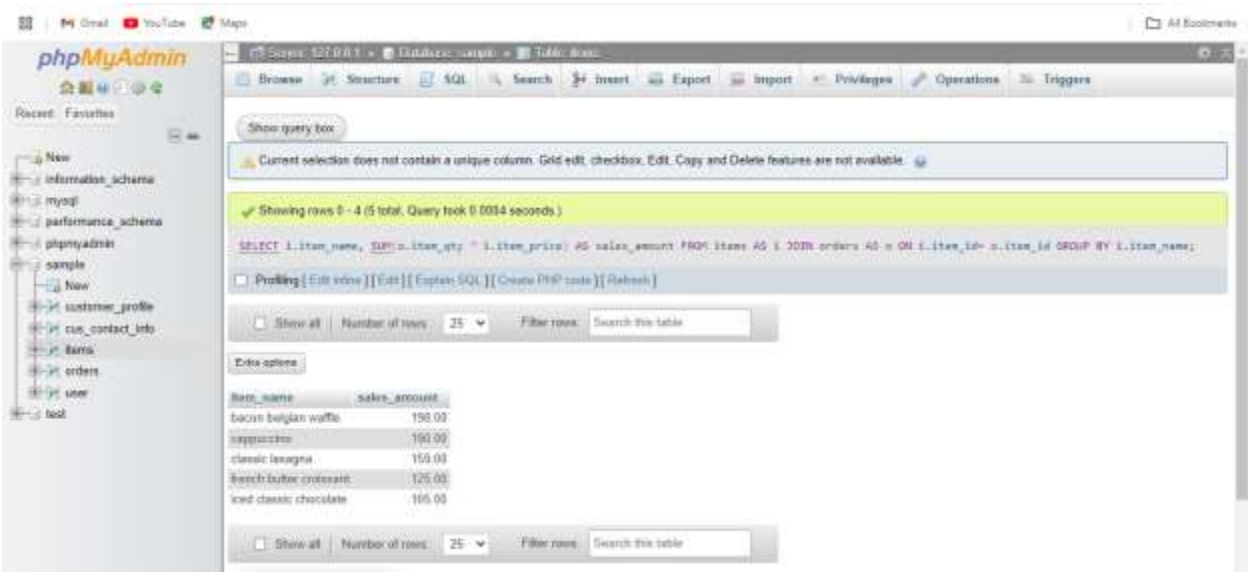
6. DISPLAYING ITEMS HAVING MORE THAN 0 SALES:

| Item name               | total_qty_sold |
|-------------------------|----------------|
| bacon belgian waffle    | 2              |
| cappuccino              | 2              |
| classic leagna          | 1              |
| french butter croissant | 1              |
| loaf classic chocolate  | 1              |

Here, we displayed all the items that contains more than 0 sales using the HAVING. This enables to filter out informations particularly the items that contains more than 0 sales.

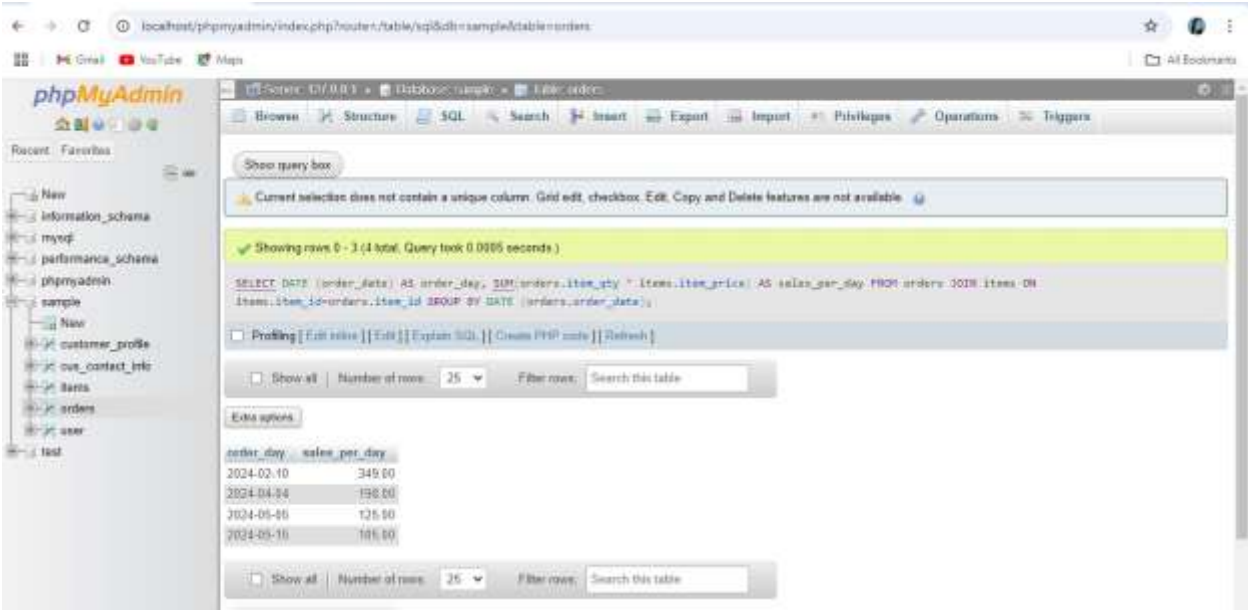


7.DISPLAY SALES



We printed the sales per item. In this case we only needed to print the item name along with the sales amount. Through this, we were able to visualize the total income that the sellers receive. Hence, being able to narrow down the information into just two columns allowed us to easily analyze how the sales were doing per item, which enables us to identify which among the products are mostly patronized by customers.

8. DISPLAYING THE TOTAL SALES PER DAY:



Here, we displayed the order date along with the sales per day, allowing us to visualize and easily identify the total number of sales per day.





9. DISPLAYING THE TOTAL SALES AND QUANTITY PER ITEMS AND DAY:

Showing rows 0 - 4 (5 total. Query took 0.0005 seconds.)

```
SELECT items.item_name, SUM(items.item_price * orders.item_qty) AS total_sales, SUM(orders.item_qty) AS quantity_per_item FROM orders JOIN items ON items.item_id = orders.item_id GROUP BY items.item_name;
```

| Item name              | total_sales | quantity_per_item |
|------------------------|-------------|-------------------|
| bacon belgian waffle   | 190.00      | 2                 |
| cappuccino             | 190.00      | 2                 |
| classic lasagna        | 150.00      | 1                 |
| leech butter croissant | 125.00      | 1                 |
| iced classic chocolate | 105.00      | 1                 |

Here, we displayed the item name along with quantity per item and the total sales, visualizing and displaying a more detailed perspective of the items, quantity and item sales.

10. DISPLAYING THE TOTAL SALES TOTAL QUANTITY SOLD PER ITEM AND DAY

Showing rows 0 - 4 (5 total. Query took 0.0005 seconds.)

```
SELECT items.item_name, SUM(items.item_price * orders.item_qty) AS Total_sales, DATE(orders.order_date) AS Order_day, SUM(orders.item_qty) AS Total_qty FROM orders JOIN items ON items.item_id = orders.item_id GROUP BY items.item_name, DATE(orders.order_date) ORDER BY 'Order_day' DESC;
```

| Item name              | Total_sales | Order_day  | Total_qty |
|------------------------|-------------|------------|-----------|
| cappuccino             | 190.00      | 2024-02-10 | 2         |
| classic lasagna        | 150.00      | 2024-02-10 | 1         |
| bacon belgian waffle   | 190.00      | 2024-04-04 | 2         |
| leech butter croissant | 125.00      | 2024-05-05 | 1         |
| iced classic chocolate | 105.00      | 2024-05-15 | 1         |

11. DISPLAYING THE TOTAL SALES AND TOTAL QUANTITY SOLD PER FULLNAME

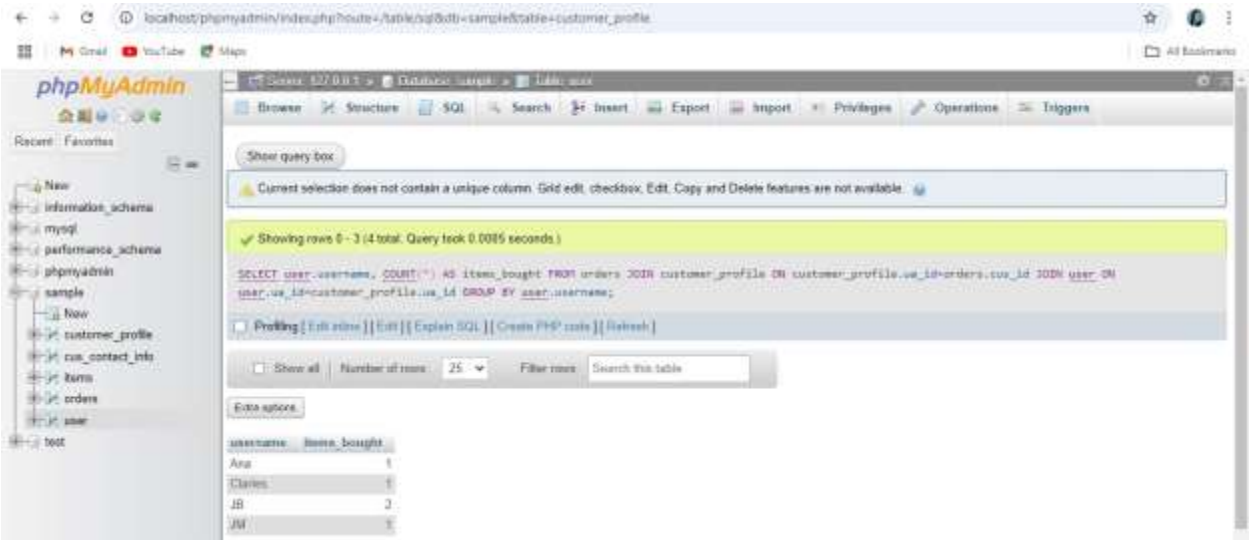
Showing rows 0 - 3 (4 total. Query took 0.0006 seconds.)

```
SELECT customer_profile.name AS fullname, SUM(orders.item_qty * items.item_price) AS total_sales, SUM(orders.item_qty) AS total_qty_sold FROM orders JOIN items ON items.item_id = orders.item_id JOIN customer_profile ON customer_profile.cus_id = orders.cus_id GROUP BY customer_profile.name;
```

| fullname | total_sales | total_qty_sold |
|----------|-------------|----------------|
| Ano      | 105.00      | 1              |
| Claves   | 90.75       | 1              |
| JB       | 290.00      | 4              |
| JM       | 95.00       | 1              |

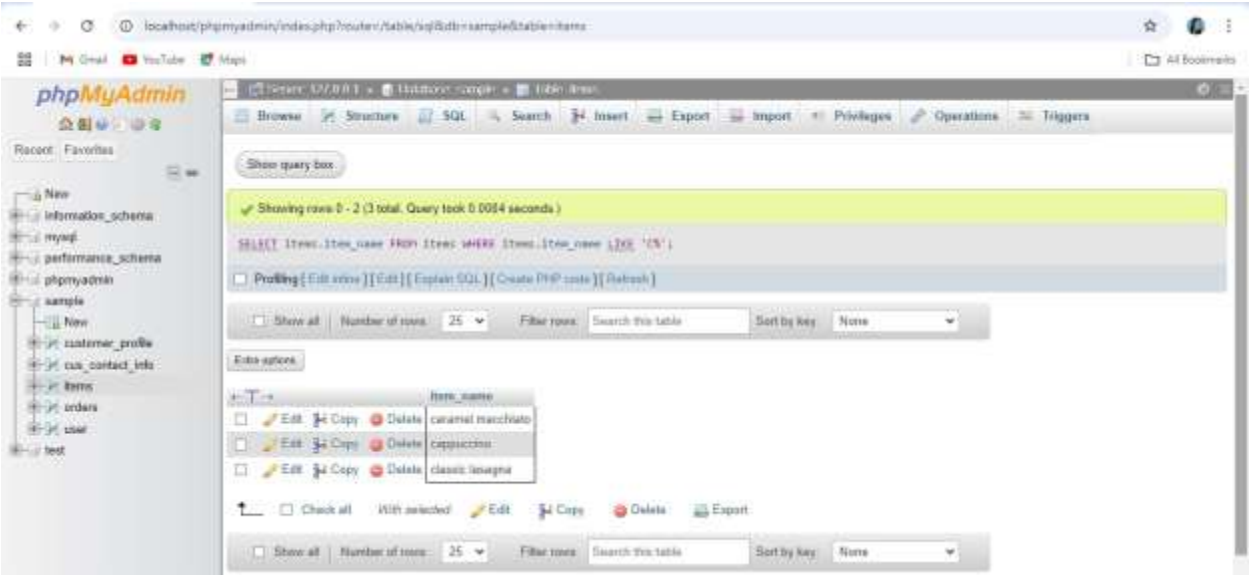


12. DISPLAYING THE NUMBERR OF ITEMS BROUGHT BY CUSTOMERS PROFILE USING COUNT (\*)



Here, we displayed the number of items purchased by each customer username through using the COUNT (\*) function in the SQL Query, allowing us to count and identify the total number of purchases associated with each customer/user

13. THE USE OF WILDCARD:



In this case, we used the wildcards to display the items from particular string starting. Additionally, we also used the SQL LIKE operator together with a wildcard character in order to filter out the result, to showcase only the items that matches and corresponds with our search criteria. Through this, displaying and picking relevant items are made easier.