

PA 2 Grocery Store

In this assignment, Flask in integration with JavaScript, HTML for front end and MySQL for backend are used to create the following locally hosted website.

A customer can add his name and continue to query for products through the drop down box given. This is done so that the customer can easily identify what products are available in the store and how they are exactly spelled. Once the customer decides on a product they're interested, the backend database is queried and it's relevant details are fetched and the price is auto filled, according to the quantity the total amount is calculated(this calculation is done in JavaScript). A customer can further add products to their cart by clicking the button "Add More". Their grand total is calculated in the last total section.

After adding all the items, customer can choose to either remove some of the items from the cart or proceed to "Place Order". By clicking Place Order, the details of the customer and products they ordered will be recorded in the backend database.

If the customer places an order without choosing any products from the drop down, nothing will be added to the database, as per the coordination protocol, since it is rejected from the server side.

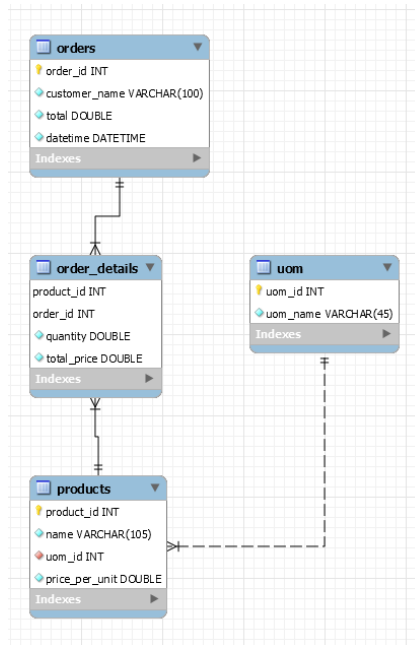
A press of a button will invoke HTML, which is run by JavaScript, that is connected to backend MySQL workbench via Python Flask which has the server.

Steps to run:

1. Run sql_connection.py with appropriate information.
2. Run server.py
3. Run order.html

Following is the image of the website:

The screenshot displays the 'GROCERY STORE' web application. At the top right, there is a text input field labeled 'Customer Name'. Below this, on the left side, is a button labeled 'Add More'. The main form area contains a 'Product' dropdown menu with '--Select--' as the current selection. Below the dropdown, the 'Price' is displayed as '0.0' in a text box. The 'Quantity' is shown as '1' in a spinner box. The 'Total : \$' is displayed in a text box, followed by a 'Remove' button. Below these fields, the word 'Total' is written in bold, followed by a text box showing '0.0' and a dollar sign '\$'. At the bottom left, there is a 'Place Order' button.



EER Diagram

GROCERY STORE

ahmed

[Add More](#)

Product :

Price :

Quantity :

Total : \$ [Remove](#)

Product :

Price :

Quantity :

Total : \$ [Remove](#)

Product :

Price :

Quantity :

Total : \$ [Remove](#)

Total

\$

[Place Order](#)

Adding items to cart

SCHEMAS

Filter objects

grocery_store

- Tables
 - order_details
 - orders
 - products
 - uom
- Views
- Stored Procedures
- Functions

Administration Schemas

Information

Table: **order_details**

Columns:

- product_id** int PK
- order_id** int PK
- quantity double
- total_price double

1 • `SELECT * FROM grocery_store.orders;`

Result Grid

order_id	customer_name	total	datetime
41	John	100	2022-03-18 22:05:30
42	Adams	1550	2022-03-18 22:06:00
43	Watson	250	2022-03-18 22:06:22
NULL	NULL	NULL	NULL

3 different customers whose order id's are different made total purchase of and the timestamp

SCHEMAS

Filter objects

grocery_store

- Tables
 - order_details
 - orders
 - products
 - uom
- Views
- Stored Procedures
- Functions

Administration Schemas

Information

Table: **order_details**

Columns:

- product_id** int PK
- order_id** int PK
- quantity double
- total_price double

1 • `SELECT * FROM grocery_store.order_details;`

Result Grid

product_id	order_id	quantity	total_price
9	41	2	40
11	41	3	60
5	42	1	30
8	42	4	1200
30	42	4	320
7	43	3	210
29	43	4	40
NULL	NULL	NULL	NULL

Products and their quantity bought by the aforementioned customers

SCHEMAS

Filter objects

- grocery_store
 - Tables
 - order_details
 - orders
 - products
 - uom
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Information

Table: order_details

Columns:

product_id	int PK
order_id	int PK
quantity	double
total_price	double

1 • `SELECT * FROM grocery_store.products;`

Limit to 2000 rows

Result Grid

	product_id	name	uom_id	price_per_unit
▶	2	moong dal	1	100
	3	tooth paste	2	30
	4	soap	2	20
	5	banana	2	30
	7	onions	2	70
	8	apple	2	300
	9	face mask	1	20
	11	okra	1	20
	12	spinach	1	34.5
	24	dal	2	2
	28	potatoes	1	10
	29	rice	1	10
	30	shrimp	1	80
•	NULL	NULL	NULL	NULL

All the products available and their price per unit

SCHEMAS

Filter objects

- grocery_store
 - Tables
 - order_details
 - orders
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
 - products
 - uom
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Information

Table: orders

Columns:

order_id	int AI PK
customer_name	varchar(100)
total	double
datetime	datetime

1 • `SELECT * FROM grocery_store.uom;`

Limit to 2000 rows

Result Grid

	uom_id	uom_name
▶	1	each
	2	kg
•	NULL	NULL

Not all items can be purchased in kg or individually