WENZHOU-KEAN UNIVERSITY

Final Project

CPS3740: Database Management System

Student Name:	Huang Yawen
Student ID:	1194921
Lecture:	Dr. Hemn Barzan Abdalla
Section	W02

Fall 2023-2024

Database Management System CPS3740

Final Project documentation Deadline: 12-04-2023

- To answers questions below first you need to create a **schema** which contain at least **four tables**
- Include the constraints like **NOT NULL, UNIQUE and CHECK** constraints in all the table.
- Insert minimum **three** records in each table.
- Retrieve the data using operators **(in, between and like).**
- **Three queries** must be there from **SET operators (Union, intersect)**
- **Three join queries** must be included out of which **three** must be outer joins.
- **Three nested queries** must be included (Inner Query and Outer Query).
- **Two queries** must be included using **group by and having.**
- **Minimum two views**, which combined of two or three tables, must be included.
- **One cursor and trigger** must be included.
- Create an **application** for your MySQL schema. (Python)
- Must have the **main form page**.
- Each table must have (**continues** form with **single** form).
- For two tables should have a **report**.
- The user when open the database must directly show the **main form.**
- Give the summary for your project plus what you learned in this semester, must be less than 300 words.

Instructions

- Create a word document which includes details and the screen shots of all the queries as said the above.
- The first page that document contains the information about the student like name, ID Number, with section number.
- Use the attached template to prepare your final project document (**Don't include Normalization).**

Answers:

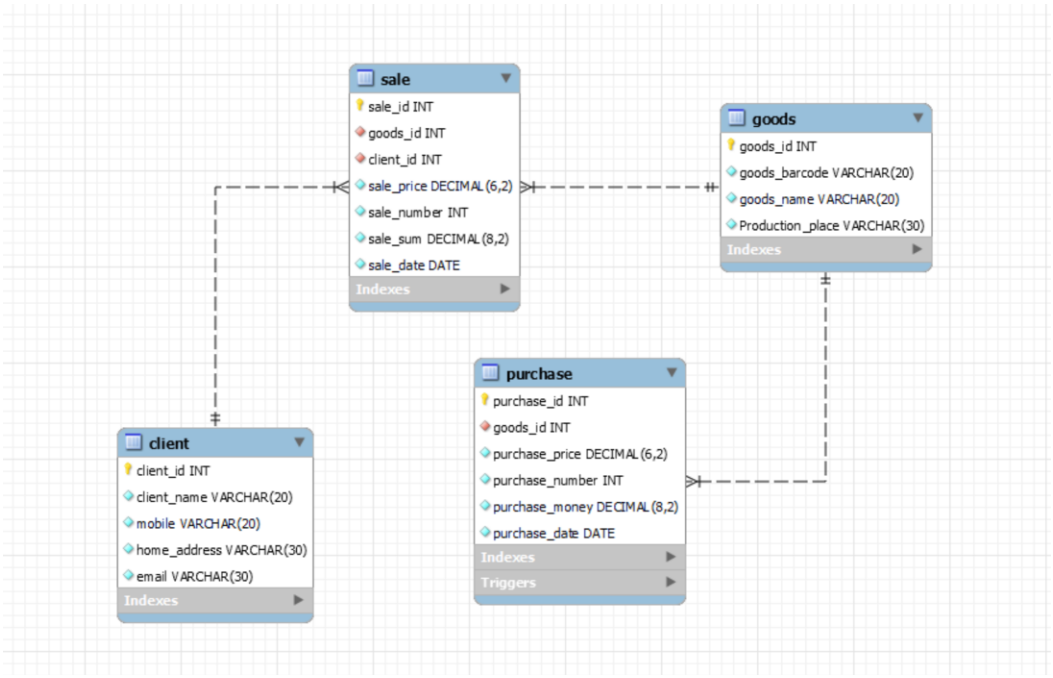
1. To answers questions below first you need to create a **schema** which contain at least **four tables**
2. Include the constraints like **NOT NULL, UNIQUE and CHECK** constraints in all the table.
3. Insert minimum **three** records in each table.

The schema is called project.

```
USE project;

CREATE TABLE `client`(
    client_id INT(10) NOT NULL,
    client_name VARCHAR(20) NOT NULL,
    mobile VARCHAR(20) NOT NULL,
    home_address VARCHAR(30) NOT NULL,
    email VARCHAR(30) NOT NULL,
    PRIMARY KEY(client_id),
    UNIQUE(mobile)
);

INSERT INTO `client` VALUES (1, 'huang', '111222333', 'daxue Road', 'huang@hi.com');
INSERT INTO `client` VALUES (2, 'li', '444555666', 'daxue Road', 'li@hi.com');
INSERT INTO `client` VALUES (3, 'niu', '777888999', 'daxue Road', 'niu@hi.com');


CREATE TABLE `goods`(
    goods_id INT(10) NOT NULL,
    goods_barcode VARCHAR(20) NOT NULL,
    goods_name VARCHAR(20) NOT NULL,
    Production_place VARCHAR(30) NOT NULL,
    PRIMARY KEY(goods_id)
);

INSERT INTO `goods` VALUES (1, '1111111', 'milk', 'Beijing');
INSERT INTO `goods` VALUES (2, '2222222', 'soap', 'Shanghai');
INSERT INTO `goods` VALUES (3, '3333333', 'banana', 'Guang xi');

CREATE TABLE `purchase`(
    purchase_id INT(10) NOT NULL,
    goods_id INT(10) NOT NULL,
    purchase_price DECIMAL(6,2) NOT NULL CHECK( purchase_price >= 0),
    purchase_number INT NOT NULL CHECK( purchase_number >= 0),
    purchase_money DECIMAL(8,2) GENERATED ALWAYS AS (purchase_price * purchase_number) STORED NOT NULL,
    purchase_date DATE NOT NULL,
    PRIMARY KEY(purchase_id),
    FOREIGN KEY (goods_id) REFERENCES goods(goods_id)
);

INSERT INTO `purchase` VALUES (1, 3, 3.00, 3, DEFAULT,'2023-12-01');
INSERT INTO `purchase` VALUES (2, 2, 12.00, 2, DEFAULT, '2023-12-02');
INSERT INTO `purchase` VALUES (3, 1, 18.00, 5, DEFAULT, '2023-12-03');
```

```sql
CREATE TABLE `sale`(
    sale_id INT(10) NOT NULL,
    goods_id INT(10) NOT NULL,
    client_id INT(10) NOT NULL,
    sale_price DECIMAL(6,2) NOT NULL CHECK( sale_price >= 0),
    sale_number INT NOT NULL CHECK( sale_number >= 0),
    sale_sum DECIMAL(8,2) GENERATED ALWAYS AS (sale_price * sale_number) STORED NOT NULL,
    sale_date DATE NOT NULL,
    PRIMARY KEY(sale_id),
    FOREIGN KEY (goods_id) REFERENCES goods(goods_id),
    FOREIGN KEY (client_id) REFERENCES client(client_id)
);
```

The diagram for these four tables:



The table is display like:

client_id	client_name	mobile	home_address	email
1	huang	111222333	daxue Road	huang@hi.com
2	li	444555666	daxue Road	li@hi.com
3	niu	777888999	daxue Road	niu@hi.com

(client)

goods_id	goods_barcode	goods_name	Production_place
1	1111111	milk	Beijing
2	2222222	soap	Shanghai
3	3333333	banana	Guang xi

(goods)

purchase_id	goods_id	purchase_price	purchase_number	purchase_money	purchase_date
1	3	3.00	3	9.00	2023-12-01
2	2	12.00	2	24.00	2023-12-02
3	1	18.00	5	90.00	2023-12-03

(purchase)

sale_id	goods_id	client_id	sale_price	sale_number	sale_sum	sale_date
1	3	1	4.00	1	4.00	2023-12-11
2	2	2	15.00	2	30.00	2023-12-12
3	1	3	22.00	1	22.00	2023-12-13

(sale)

4. Retrieve the data using operators **(in, between and like).**

IN:

```
SELECT * FROM `purchase`
WHERE goods_id IN (1, 2);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

purchase_id	goods_id	purchase_price	purchase_number	purchase_money	purchase_date
3	1	18.00	5	90.00	2023-12-03
2	2	12.00	2	24.00	2023-12-02

BETWEEN:

```
SELECT * FROM `purchase`
WHERE purchase_price BETWEEN 10.00 AND 20.00;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

purchase_id	goods_id	purchase_price	purchase_number	purchase_money	purchase_date
2	2	12.00	2	24.00	2023-12-02
3	1	18.00	5	90.00	2023-12-03

LIKE:

```
SELECT * FROM `clients`
WHERE email LIKE '%@hi.com';
```

client_id	client_name	mobile	home_address	email
1	huang	111222333	daxue Road	huang@hi.com
2	li	444555666	daxue Road	li@hi.com
3	niu	777888999	daxue Road	niu@hi.com

5. **Three queries** must be there from **SET operators (Union, intersect)**

Using **UNION** to combine records from the **purchase** and **sale** tables:

```
SELECT purchase_id, goods_id, purchase_price, purchase_number, purchase_money, purchase_date
FROM `purchase`
UNION
SELECT sale_id, goods_id, sale_price, sale_number, sale_sum, sale_date
FROM `sale`;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

purchase_id	goods_id	purchase_price	purchase_number	purchase_money	purchase_date
1	3	3.00	3	9.00	2023-12-01
2	2	12.00	2	24.00	2023-12-02
3	1	18.00	5	90.00	2023-12-03
1	3	4.00	1	4.00	2023-12-11
2	2	15.00	2	30.00	2023-12-12
3	1	22.00	1	22.00	2023-12-13

Achieving **INTERSECT** to find goods that appear both in purchases and sales by:

```
SELECT DISTINCT p.goods_id
FROM purchase p
INNER JOIN sale s ON p.goods_id = s.goods_id;
```

goods_id
1
2
3

Using **UNION** to retrieve a distinct list of goods IDs from both the purchase and sale tables

```
SELECT goods_id FROM purchase
UNION
SELECT goods_id FROM sale;
```

goods_id
1
2
3

6. **Three join queries** must be included out of which three must be outer joins.

LEFT OUTER JOIN:

```
SELECT goods.goods_id, goods_name, purchase_id, purchase_price, purchase_number, purchase_money, purchase_date
FROM goods
LEFT OUTER JOIN purchase ON goods.goods_id = purchase.goods_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

goods_id	goods_name	purchase_id	purchase_price	purchase_number	purchase_money	purchase_date
1	milk	3	18.00	5	90.00	2023-12-03
2	soap	2	12.00	2	24.00	2023-12-02
3	banana	1	3.00	3	9.00	2023-12-01

RIGHT OUTER JOIN:

```
SELECT sale.sale_id, client_name, sale_price, sale_number, sale_sum, sale_date
FROM client
RIGHT OUTER JOIN sale ON client.client_id = sale.client_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sale_id	client_name	sale_price	sale_number	sale_sum	sale_date
1	huang	4.00	1	4.00	2023-12-11
2	li	15.00	2	30.00	2023-12-12
3	niu	22.00	1	22.00	2023-12-13

LEFT OUTER JOIN: (another one)

```
37 •  SELECT client.client_id, client_name, sale_id, sale_price, sale_number, sale_sum, sale_date
38    FROM client
39    LEFT OUTER JOIN sale ON client.client_id = sale.client_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

client_id	client_name	sale_id	sale_price	sale_number	sale_sum	sale_date
1	huang	1	4.00	1	4.00	2023-12-11
2	li	2	15.00	2	30.00	2023-12-12
3	niu	3	22.00	1	22.00	2023-12-13

7. **Three nested queries** must be included (Inner Query and Outer Query).

```
42 •  SELECT SUM(sale_sum) AS total_sale_amount
43    FROM sale
44    WHERE client_id = (
45        SELECT client_id
46        FROM client
47        WHERE client_name = 'li'
48    );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sale_amount
30.00

```
50 •  SELECT client_id
51    FROM client
52    WHERE client_name = 'huang';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sale_amount
30.00

```
54 •  SELECT client_name, email
55    FROM client
56    WHERE client_id IN (
57        SELECT DISTINCT client_id
58        FROM sale
59    );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

client_name	email
huang	huang@hi.com
li	li@hi.com
niu	niu@hi.com

8. **Two queries** must be included using **group by and having.**

GROUP BY:

```
61 •  SELECT g.goods_id, g.goods_name, SUM(p.purchase_money) AS total_purchase_money
62    FROM goods g
63    JOIN purchase p ON g.goods_id = p.goods_id
64    GROUP BY g.goods_id, g.goods_name;
65
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

goods_id	goods_name	total_purchase_money
3	banana	9.00
2	soap	24.00
1	milk	90.00

HAVING:

```
66 •  SELECT c.client_id, c.client_name, SUM(s.sale_sum) AS total_sales
67    FROM client c
68    JOIN sale s ON c.client_id = s.client_id
69    GROUP BY c.client_id
70    HAVING total_sales > 10;
71
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

goods_id	goods_name	total_purchase_money
3	banana	9.00
2	soap	24.00
1	milk	90.00

9. **Minimum two views**, which combined of two or three tables, must be included.

(A view is a virtual table based on the result of a SELECT query.)

VIEW1:

```
72 •  CREATE VIEW client_purchase_view AS
73    SELECT c.client_id, c.client_name, p.purchase_id, p.purchase_price, p.purchase_number, p.purchase_money,
74           g.goods_id, g.goods_name
75    FROM client c
76    JOIN sale s ON c.client_id = s.client_id
77    JOIN purchase p ON s.goods_id = p.goods_id
78    JOIN goods g ON p.goods_id = g.goods_id;
79
80 •  SELECT * FROM client_purchase_view;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

client_id	client_name	purchase_id	purchase_price	purchase_number	purchase_money	goods_id	goods_name
1	huang	1	3.00	3	9.00	3	banana
2	li	2	12.00	2	24.00	2	soap
3	niu	3	18.00	5	90.00	1	milk

This view provides information about client along with details of their purchases and the corresponding goods.

VIEW2

```
82 •  CREATE VIEW sale_goods_client_view AS
83    SELECT s.sale_id, s.sale_price, s.sale_number, s.sale_sum,
84           g.goods_id, g.goods_name, g.Production_place,
85           c.client_id, c.client_name, c.mobile, c.home_address, c.email
86    FROM sale s
87    JOIN goods g ON s.goods_id = g.goods_id
88    JOIN client c ON s.client_id = c.client_id;
89
90 •  SELECT * FROM sale_goods_client_view;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sale_id	sale_price	sale_number	sale_sum	goods_id	goods_name	Production_place	client_id	client_name	mobile	home_address	email
1	4.00	1	4.00	3	banana	Guang xi	1	huang	111222333	daxue Road	huang@hi.com
2	15.00	2	30.00	2	soap	Shanghai	2	li	444555666	daxue Road	li@hi.com
3	22.00	1	22.00	1	milk	Beijing	3	niu	777888999	daxue Road	niu@hi.com

This view provides information about sales along with details of the corresponding goods and clients involved in those sales.

10. **<u>One cursor and trigger must be included.</u>**

Cursor:

```
92    DELIMITER //
93 •  CREATE PROCEDURE update_purchase_total()
94    BEGIN
95        DECLARE done BOOLEAN DEFAULT FALSE;
96        DECLARE purchase_id_val INT(10);
97        DECLARE purchase_price_val DECIMAL(6,2);
98        DECLARE purchase_number_val INT;
99
100       DECLARE purchase_cursor CURSOR FOR
101           SELECT purchase_id, purchase_price, purchase_number
102           FROM purchase;
103
104       DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
105
106       OPEN purchase_cursor;
```

```
108       read_loop: LOOP
109           FETCH purchase_cursor INTO purchase_id_val, purchase_price_val, purchase_number_val;
110
111           IF done THEN
112               LEAVE read_loop;
113           END IF;
114
115           -- Perform any calculations or updates here
116           -- For example, you can update the total in the purchase table
117           UPDATE purchase
118           SET purchase_money = purchase_price_val * purchase_number_val
119           WHERE purchase_id = purchase_id_val;
120       END LOOP;
121
122       CLOSE purchase_cursor;
123   END //
124   DELIMITER ;
```

In the example procedure **update_purchase_total**, a **cursor** is declared to iterate through the records in the **purchase** table. Inside the loop, you can perform actions on each record. It updates the **purchase_money** column based on the product of **purchase_price** and **purchase_number**.

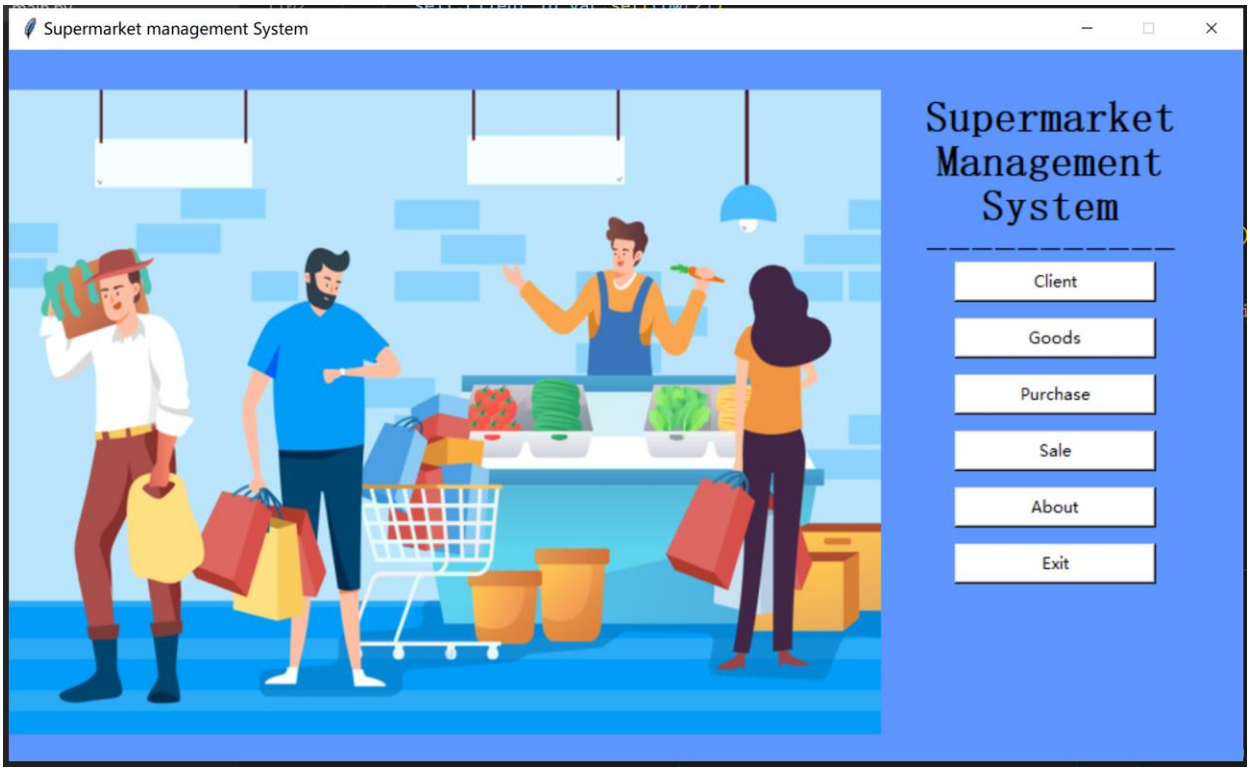
Trigger:

```
128     -- Trigger
129     DELIMITER //
130     CREATE TRIGGER update_purchase_money
131     AFTER INSERT ON purchase
132     FOR EACH ROW
133     BEGIN
134       UPDATE purchase SET purchase_money = NEW.purchase_price * NEW.purchase_number WHERE purchase_id = NEW.purchase_id;
135     END;
136     //
137     DELIMITER ;
```

The trigger **update_purchase_money** automatically calculates and updates the **purchase_money** column in the **purchase** table after a new record is inserted. It uses the **AFTER INSERT** event and sets the **purchase_money** based on the product of the newly inserted **purchase_price** and **purchase_number**

11. Create an **application** for your MySQL schema. (Python)
12. Must have the **main form page**.
13. The user when open the database must directly show the **main form.**
14. Each table must have (**continues** form with **single** form).
15. For two tables should have a **report**.
16. The user when open the database must directly show the **main form.**

Main form:



Code of the main form (includes the variables we need to use)

```python
from tkinter import *
from PIL import ImageTk, Image
import tkinter as tk
from tkinter import ttk
import pandas as pd
import mysql.connector
from tkinter import messagebox
import pymysql

WINDOW_WIDTH = 920
WINDOW_HEIGHT = 530
TOP_WIDTH = 800
TOP_HEIGHT = 480
# Overall background color
BG_COLOR = '#5E95FF'
# Font for label
TK_L_FONT = ('', 28, 'bold')
# Font for entry
TK_E_FONT = ('', 18, 'bold')
# Font for button
TK_B_FONT = ('', 28, 'bold')
```

```python
class TkinterMain():

    def __init__(self):
        #Initialize the window body
        self.window = Tk()
        self.window.config(background=BG_COLOR)
        self.window.title("Supermarket management System")

        # Get screen height and width
        screen_width = self.window.winfo_screenwidth()
        screen_height = self.window.winfo_screenheight()

        # Set the window height and width and center it on the screen
        x = (screen_width - WINDOW_WIDTH) / 2
        y = (screen_height - WINDOW_HEIGHT) / 2
        root_size = '%dx%d+%d+%d' % ( WINDOW_WIDTH, WINDOW_HEIGHT, x, y)
        self.window.geometry(root_size)
        # Fixed size
        self.window.wm_resizable(False, False)

        #-------- Variables ----
        self.client_id_var=StringVar()
        self.client_name_var=StringVar()
        self.mobile_var=StringVar()
        self.home_address_var=StringVar()
        self.email_var=StringVar()
```

```python

            self.goods_id_var=StringVar()
            self.goods_barcode_var=StringVar()
            self.goods_name_var=StringVar()
            self.Production_place_var=StringVar()
            self.del_var=StringVar()
            self.search_var=StringVar()
            self.co_var=StringVar() # add this variable for search combobox

            self.purchase_id_var=StringVar()
            #self.goods_id_var=StringVar()
            self.purchase_price_var=StringVar()
            self.purchase_number_var=StringVar()
            self.purchase_money_var=StringVar()
            self.purchase_date_var=StringVar()
            #self.del_var=StringVar()
            #self.search_var=StringVar()
            #self.co_var=StringVar() # add this variable for search combobox

            self.sale_id_var=StringVar()
            #self.goods_id_var=StringVar()
            #self.client_id_var=StringVar()
            self.sale_price_var=StringVar()
            self.sale_number_var=StringVar()
            self.sale_sum_var=StringVar()
            self.sale_date_var=StringVar()
```

I defined and used the method show_top_view() to show the sub windows.

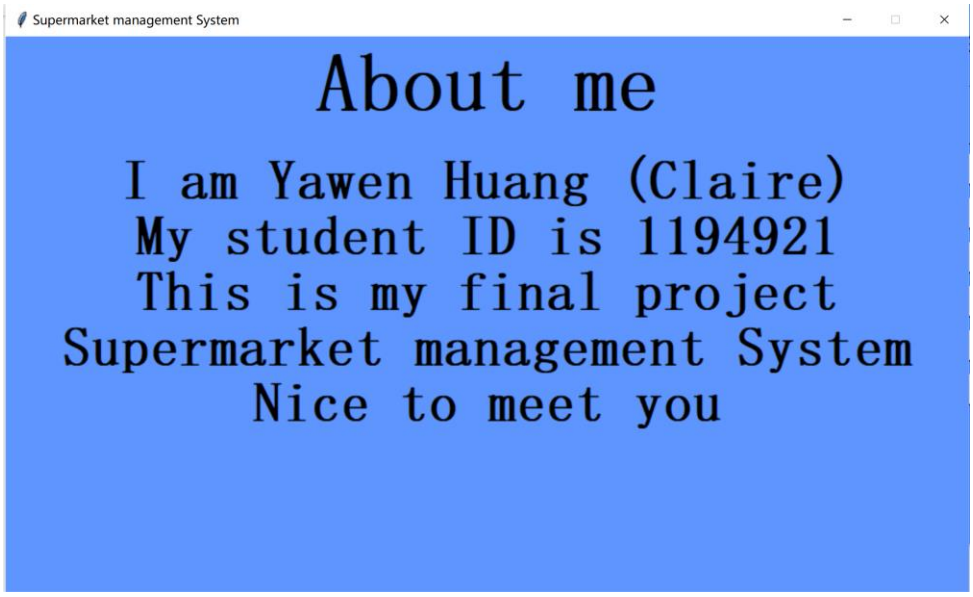
show_top_view() includes

self.top_view = Toplevel(background=BG_COLOR)

```
        # Sub window
        self.top_view = None
        #Create homepage window
        self.create_home_page_view()

    #Create homepage
    def create_home_page_view(self):
        # Home page background
        img = Image.open('bg.png').resize((650, 480))
        self.bg_png = ImageTk.PhotoImage(img)
        Label(self.window, image=self.bg_png, bd=0).place(x=0, y=30)
        # title
        Label(self.window, text='Supermarket\nManagement\nSystem\n----------',
              font=('', 24, 'bold'), bg=BG_COLOR).place(x=680, y=30)

        # Button frame
        btn_frame = Frame(self.window, bg=BG_COLOR)
        btn_frame.place(x=680, y=152, width=200, height=350)

        Button(btn_frame, text='Client', width=20,command=self.client, bg='white').pack(pady=6)
        Button(btn_frame, text='Goods', width=20,command=self.goods, bg='white').pack(pady=6)
        Button(btn_frame, text='Purchase', width=20,command=self.purchase, bg='white').pack(pady=6)
        Button(btn_frame, text='Sale', width=20,command=self.sale, bg='white').pack(pady=6)
        Button(btn_frame, text='About', width=20,command=self.about, bg='white').pack(pady=6)
        Button(btn_frame, text='Exit', width=20,command=self.quit, bg='white').pack(pady=6)
```

```
    # Pop up the sub window
    def show_top_view(self):
        if self.top_view is not None:
            self.top_view.destroy()
            self.top_view = None
        self.top_view = Toplevel(background=BG_COLOR)

        # Get screen height and width
        screen_width = self.window.winfo_screenwidth()
        screen_height = self.window.winfo_screenheight()

        # Set the window height and width and center it on the screen
        x = (screen_width - WINDOW_WIDTH) / 2
        y = (screen_height - WINDOW_HEIGHT) / 2
        root_size = '%dx%d+%d+%d' % ( WINDOW_WIDTH, WINDOW_HEIGHT, x, y)
        self.top_view.geometry(root_size)

        # Fixed size
        self.top_view.wm_resizable(False, False)

    # Display main Page
    def show_window(self):
        self.window.mainloop()


if __name__ == '__main__':
    m = TkinterMain()
    m.show_window()
```

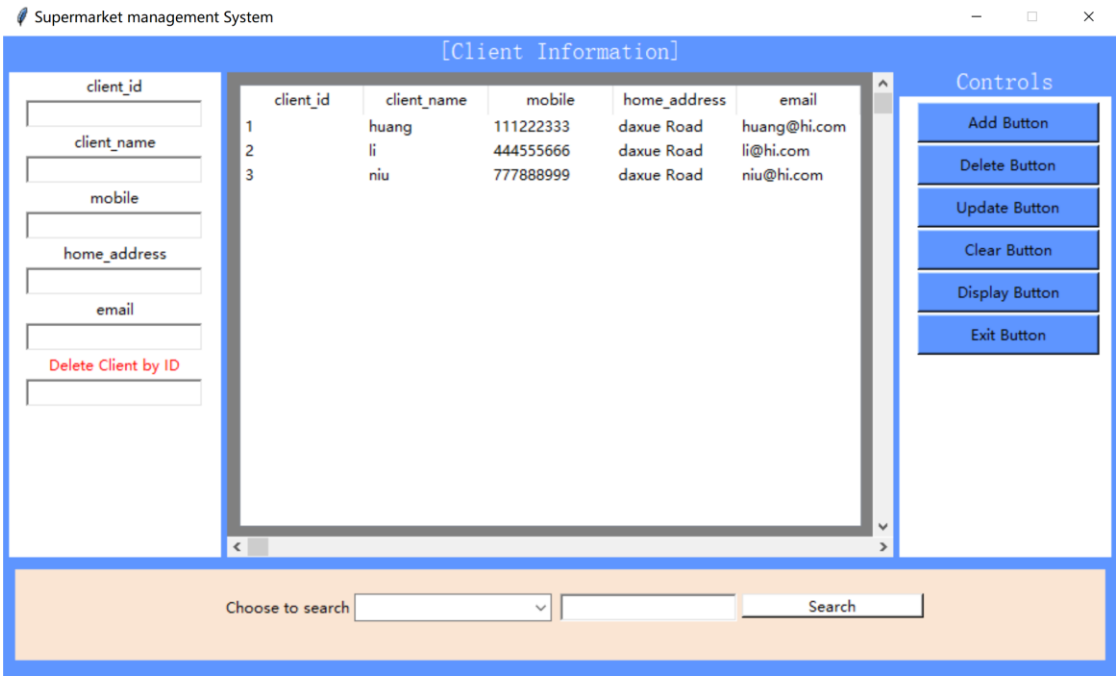
Click Button About → to see the information of me




```
def about(self):
    self.show_top_view()

    Label(self.top_view, text='About me', font=('', 60, 'bold'), bg=BG_COLOR).pack()
    info_frame = Frame(self.top_view, width=WINDOW_WIDTH, bg=BG_COLOR)
    info_frame.pack(pady=(20, 0), expand=True, fill='both')
    Label(info_frame, text='I am Yawen Huang (Claire)\nMy student ID is 1194921\nThis is my final project\nSupermarket management System\nNice to meet you',
          font=('', 40, 'bold'), bg=BG_COLOR).pack()
```

The main form links with the four tables I create in MySQL, you could enter the table interface through click the buttons:

For example, Click Button Client →



You could add, delete, update and search data by this application.

Add:

```
def addClient(self): #-- pip install PyMySQL -- need to install----
    con=pymysql.Connect(host='localhost', user='root', password='sw06010814',
                        database='project')
    cur=con.cursor()
    cur.execute("insert into client values(%s,%s,%s,%s,%s) ",(
                        self.client_id_var.get(),
                        self.client_name_var.get(),
                        self.mobile_var.get(),
                        self.home_address_var.get(),
                        self.email_var.get()
                        )
            )
    con.commit()
    self.fetch_allClient() # display all rows
    self.clearClient()# after added record need to clear all entries
    con.close()

def fetch_allClient(self):
    con=pymysql.Connect(host='localhost', database='project', user='root',
                    password='sw06010814')
    cur=con.cursor()
    cur.execute("select * from client")
    rows=cur.fetchall()
    if len (rows) !=0:
        self.client_table.delete(*self.client_table.get_children())
        for row in rows:
            self.client_table.insert("", END, values=row)
            con.commit()
        con.close()
```

Delete:

```
# ---delete client name -----
def deleteClient(self):
    con=pymysql.Connect(host='localhost', database='project', user='root',
                    password='sw06010814')
    cur=con.cursor()
    cur.execute('delete from client where client_id=%s', self.del_var.get())
    con.commit()
    self.fetch_allClient()
    con.close()
```

Update:

```
#---- Update record -----
def updateClient(self):
    con=pymysql.Connect(host='localhost', user='root', password='sw06010814',
                        database='project')
    cur=con.cursor()
    cur.execute("update client set client_name=%s, mobile=%s, home_address=%s ,email=%s where client_id=%s",(
                        self.client_name_var.get(),
                        self.mobile_var.get(),
                        self.home_address_var.get(),
                        self.email_var.get(),
                        self.client_id_var.get()
                        )
            )
    con.commit()
    self.fetch_allClient() # display all rows
    self.clearClient()# after added record need to clear all entries
    con.close()
```

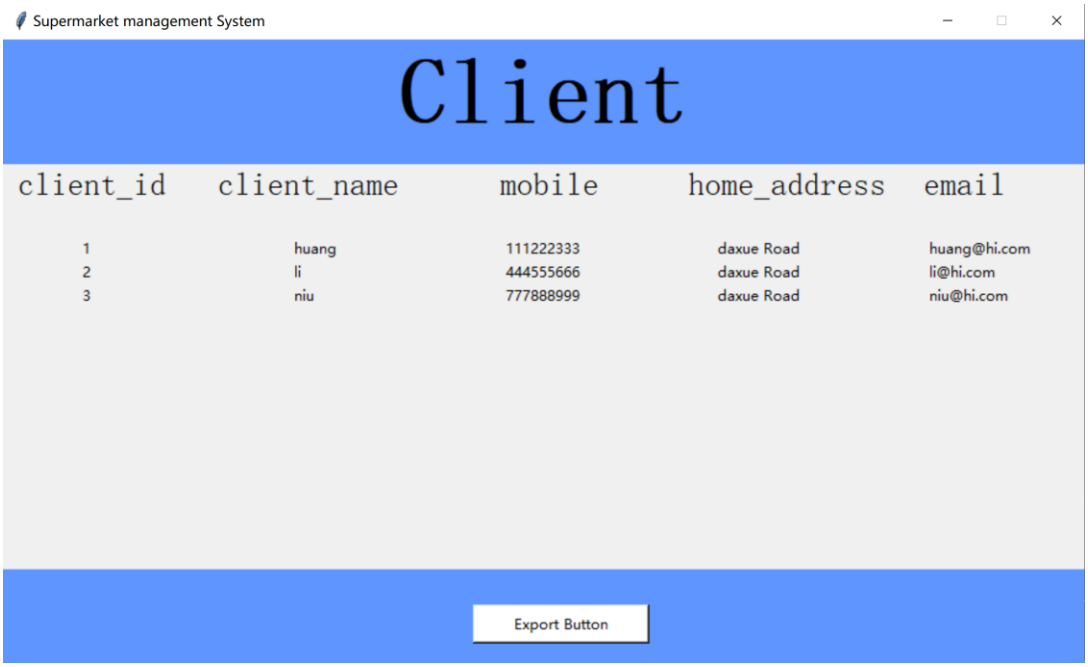
Clear:

```
# ---clear entries -----
def clearClient(self):
    self.client_id_var.set('')
    self.client_name_var.set('')
    self.mobile_var.set('')
    self.home_address_var.set('')
    self.email_var.set('')
```

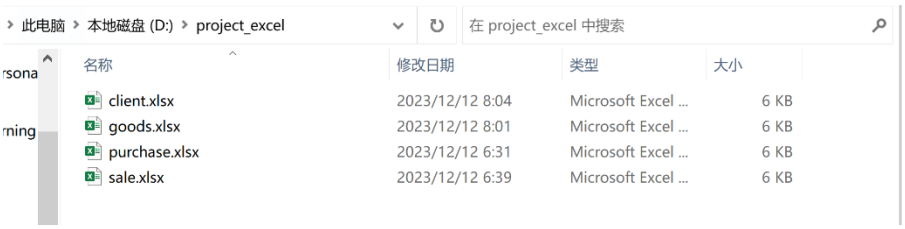
Search:

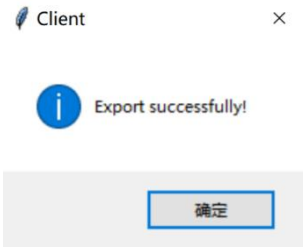
```
#---- Search -----
def searchClient(self):
    con=pymysql.Connect(host='localhost', database='project', user='root',
                        password='sw06010814')
    cur=con.cursor()
    cur.execute("select * from client where " +
    str(self.co_var.get())+" LIKE '%"+str(self.search_var.get())+"%'")
    rows=cur.fetchall()
    if len (rows) !=0:
        self.client_table.delete(*self.client_table.get_children())
        for row in rows:
            self.client_table.insert("", END, values=row)
    con.commit()
    con.close()
```

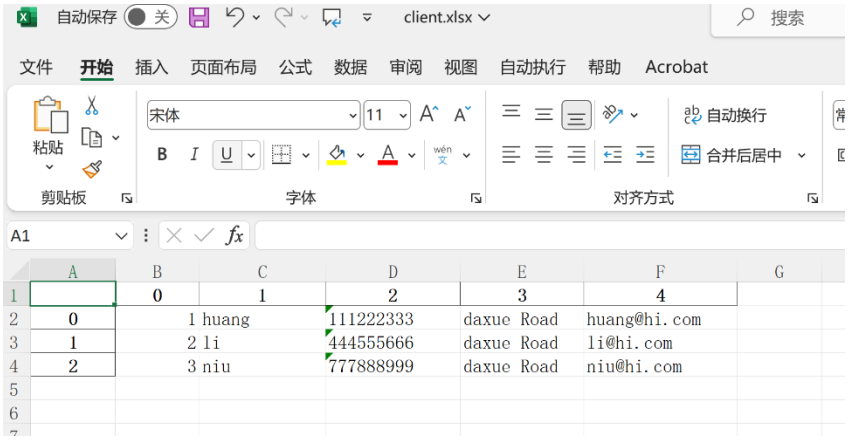
Also, you could Display the database information by click the Display Button.



And you can also export it to an excel file, just click the export Button, then the application will generate a file under 'D:\project_excel'







The method I used to display the report and export to excel:

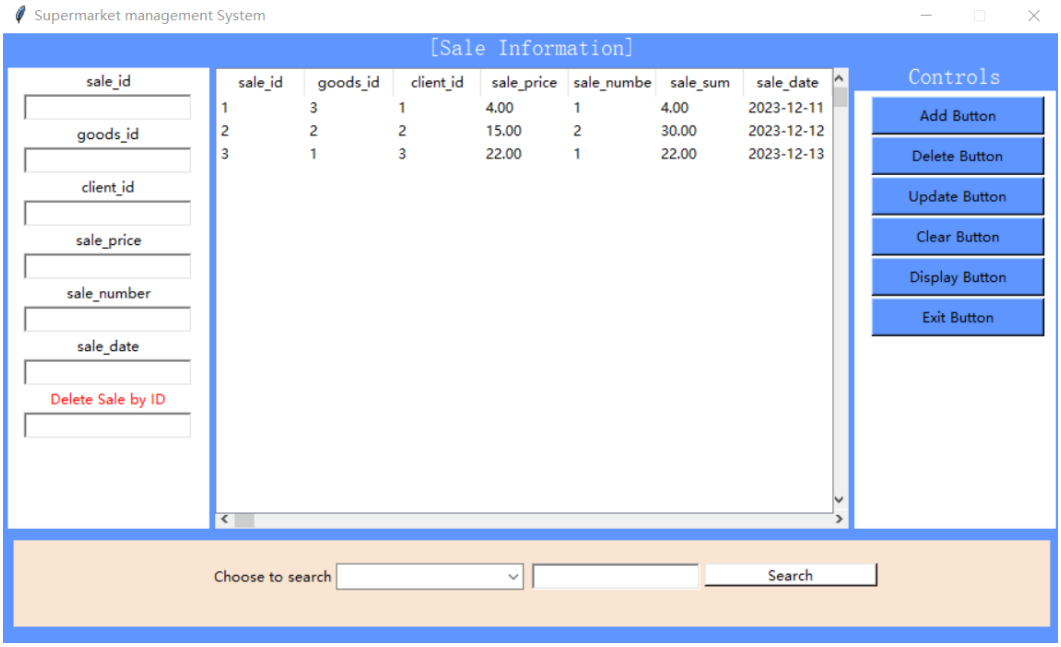
```
def displayClient(self):
    self.show_top_view()
    con=pymysql.Connect(host='localhost', user='root', password='sw06010814',
                        database='project')

    cur=con.cursor()
    cur.execute("select * from client")
    rows=cur.fetchall()


    Label(self.top_view, text='Client', font=('', 60, 'bold'), bg=BG_COLOR).pack()
    info_frame = Frame(self.top_view, width=WINDOW_WIDTH)
    info_frame.pack(pady=(20, 0), expand=True, fill='y')
    Label(info_frame, text='client_id', font=('', 20)).place(x=10, y=0)
    Label(info_frame, text='client_name', font=('', 20)).place(x=180, y=0)
    Label(info_frame, text='mobile', font=('', 20)).place(x=420, y=0)
    Label(info_frame, text='home_address', font=('', 20)).place(x=580, y=0)
    Label(info_frame, text='email', font=('', 20)).place(x=780, y=0)

    info_all_frame = Frame(info_frame)
    info_all_frame.place(x=0, y=36, width=TOP_WIDTH, height=338)
```

```
    for i, row in enumerate(rows):
        for j, value in enumerate(row):
            fund_info_label = tk.Label(info_frame, text=value)
            fund_info_label.place(x=(j * 180 + 65), y=((i + 3) * 20))

    btn_Frame= Frame(self.top_view, bg='#5E95FF')
    btn_Frame.place(x=0,y=450, width=920, height=100)
    export_btn=Button(btn_Frame, text='Export Button', bg='white', command=self.client_export_to_excel)
    export_btn.place(x=400, y=30, width=150, height=33)
```

```
def client_export_to_excel(self):
    con = mysql.connector.connect(host='localhost', user='root', password='sw06010814',
                                  database='project')

    cur = con.cursor()
    cur.execute("SELECT * FROM client")
    sqldata = cur.fetchall()
    # Send data into dataframe
    df = pd.DataFrame(data=sqldata)
    cur.close()
    # view dataframe
    print(df)
    #to_excel
    df.to_excel('D:\project_excel/client.xlsx', sheet_name='client')
    messagebox.showinfo("Client","Export successfully!")
```

Because the page for 4 table is generally consistent, I will just show 2 of them to you.

Click Button Sale →



This page is slightly different from the previous one because you don't need to input the sale_sum. The sale_sum would generate automatically.

17. Give the summary for your project plus what you learned in this semester, must be less than 300 words.

In this project, I used MySQL to create the schema and table and Python tkinter to complete the design of the application interface. I used PyMySQL to connect to the local database to add, delete, check and modify data. This project has trained me to write SQL language, and the ability to use Python tkinter to create a graphical interface. I learned how to build a database that can be interacted with, greatly benefiting me.

During the semester, I acquired skills in designing and implementing relational database schemas, ensuring data integrity through constraints, crafting intricate SQL queries with joins and set operations, and creating a primary application to interact with the database. Additionally, I garnered hands-on experience in utilizing Python tkinter to create the graphical interface and MySQL for practical applications. Furthermore, I gained theoretical understanding in areas such as database design paradigms, transactions, and concurrency control. This information will be extremely useful in my future database design and optimization work.