Phase 9

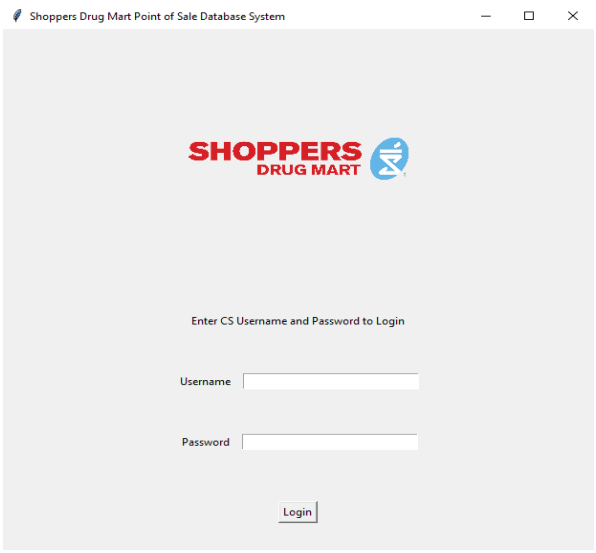
Toronto Metropolitan University

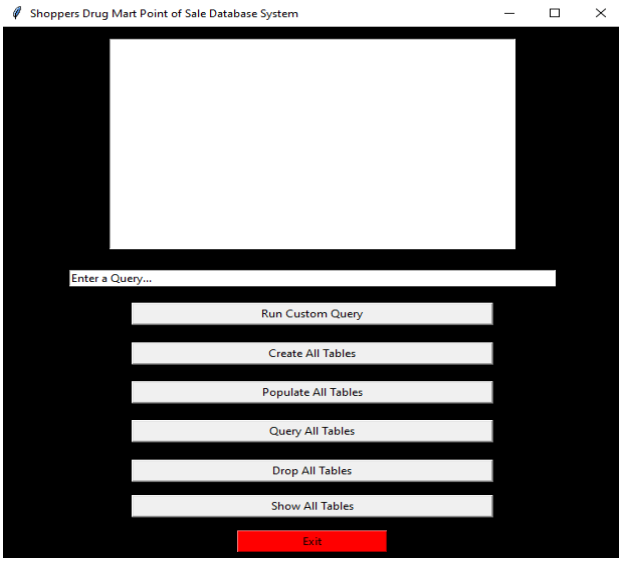
Simon Lin (501103322),
Dylan Ha (501056670),
Enes Polat (501061594)

CPS510 - Database Systems

Point of Sale System for Shopper Drug Marts

Python Implementation of Shoppers DBMS (all files included in A9 submission)





https://github.com/lin-simon/POSShoppersA9/readme.md

Point of Sale Shoppers Drug Mart DBMS

Simon Lin (501103322), Dylan Ha (501056670) and Enes Polat (501061594)

This is a Point of Sale (POS) System for Shoppers Drug Mart UI, developed for CPS510 - Database Systems Assignment 9. The system allows for management of products, inventory, transactions, customers, receipts, and employees in a retail environment. It is built with Python using Tkinter for the UI, and connects through Oracle SQL for database operations.

Requirements:

- Python 3
- tkinter (run pip install tk)
- cx_Oracle (run pip install cx_Oracle)
- PIL (Pillow, run pip install pillow)

Once installed, start CS VPN connection and run main.py and login with your proper cs credentials to access the DB.

Below are some sample queries you can run using the custom query input (note: cx_oracle syntax does not require semicolons at the end of statements):

INSERT INTO product (product_id, category, product_name, price, shelf_quantity) VALUES (1, 'Fruit', 'Apple', 0.99, 10)

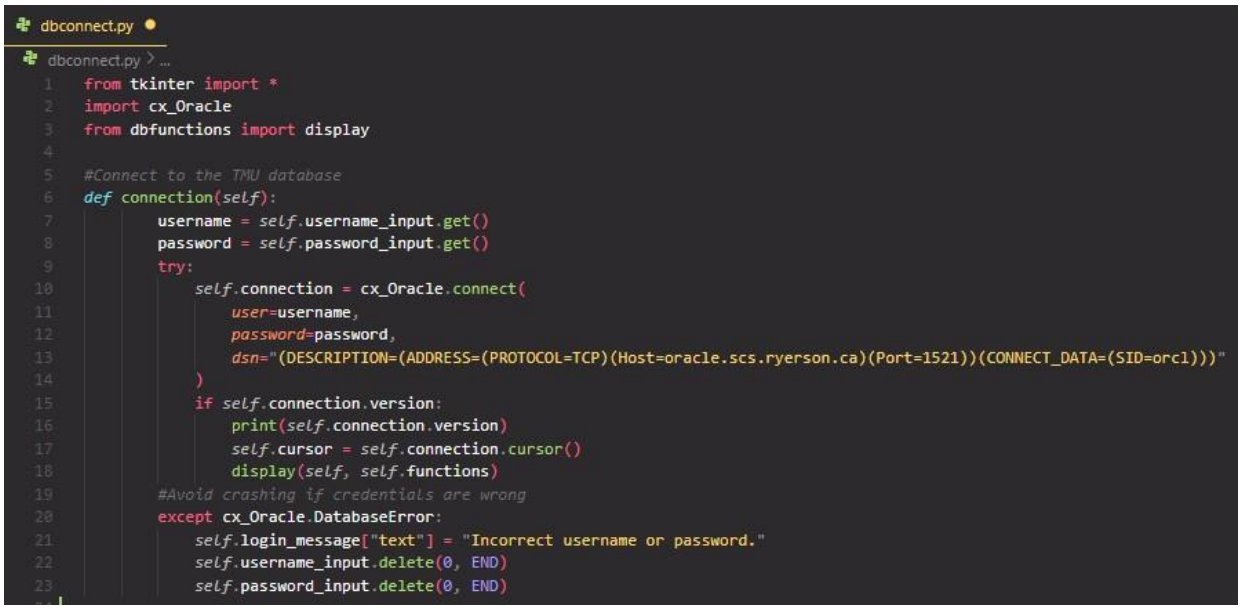
SELECT * FROM product WHERE price > 5

SELECT * FROM customer

SELECT payment_method AS "Payment_Method", SUM(total_price) AS "Total Sales" FROM receipt GROUP BY payment_method

SELECT name, 'Works as a ' || position || ' for Shoppers Drug Mart' AS "Job Description" FROM employee

dbconnect.py - For connecting UI to Database



```python
from tkinter import *
import cx_Oracle
from dbfunctions import display

#Connect to the TMU database
def connection(self):
        username = self.username_input.get()
        password = self.password_input.get()
        try:
            self.connection = cx_Oracle.connect(
                user=username,
                password=password,
                dsn="(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
            )
            if self.connection.version:
                print(self.connection.version)
                self.cursor = self.connection.cursor()
                display(self, self.functions)
        #Avoid crashing if credentials are wrong
        except cx_Oracle.DatabaseError:
            self.login_message["text"] = "Incorrect username or password."
            self.username_input.delete(0, END)
            self.password_input.delete(0, END)
```

dbfunctions.py - Create, Drop, Populate, Querying tables and exiting

dbconnect.py • dbfunctions.py ×

dbfunctions.py > ...

```python
from tkinter import *
import cx_Oracle

#Let user input their own custom query
def custom_query(self):
    #Get the result
    query = self.input.get()

    self.result.config(state=NORMAL)
    self.result.delete("1.0", END)
    try:
        self.cursor.execute(query)
        rows = self.cursor.fetchall()
        for row in rows:
            #Display the result
            self.result.insert(END, f"{row}\n")
    except cx_Oracle.DatabaseError:
        self.result.insert(END, "Query syntax incorrect (no semicolons)")
    #Clear field
    self.result.config(state=DISABLED)

def create_all_tables(self):
    #Commands for creating all our tables
    #Clear previous field
    self.result.config(state=NORMAL)
    self.result.delete("1.0", END)
    sql_commands = [
        """
        CREATE TABLE optimum (
            Optimum_ID NUMBER PRIMARY KEY,
            Total_Points NUMBER DEFAULT 0 CHECK (Total_Points >= 0),
            Name VARCHAR2(25) NOT NULL
        )
        """,
        """
        CREATE TABLE customer (
            Customer_ID NUMBER UNIQUE,
            Optimum_ID NUMBER REFERENCES optimum(Optimum_ID),
            Name VARCHAR2(25),
            PRIMARY KEY (Customer_ID, Optimum_ID)
        )
        """,
        """
        CREATE TABLE employee (
            Employee_ID NUMBER PRIMARY KEY,
            Position VARCHAR2(25) NOT NULL,
            Name VARCHAR2(25) NOT NULL
        )
        """,
        """
        CREATE TABLE product (
            Product_ID NUMBER PRIMARY KEY,
            Category VARCHAR2(25),
            Product_Name VARCHAR2(255),
            Price DECIMAL(10, 2) CHECK (Price >= 0),
            Shelf_Quantity NUMBER DEFAULT 0 CHECK (Shelf_Quantity >= 0)
        )
        """,
        """
        CREATE TABLE transaction (
            Transaction_ID NUMBER UNIQUE,
            Employee_ID NUMBER REFERENCES employee(Employee_ID),
            Total_Points NUMBER,
            Total_Price DECIMAL(10, 2) CHECK (Total_Price >= 0),
            Payment_Method VARCHAR2(6),
            Transaction_Date DATE,
            PRIMARY KEY (Transaction_ID, Employee_ID)
        )
        """,
        """
        CREATE TABLE receipt (
            Transaction_ID NUMBER PRIMARY KEY,
            Product_List VARCHAR2(255),
            Points_Earned NUMBER,
            Total_Price DECIMAL(18, 2) CHECK (Total_Price >= 0),
            Payment_Method VARCHAR2(6),
            Transaction_Date DATE
```

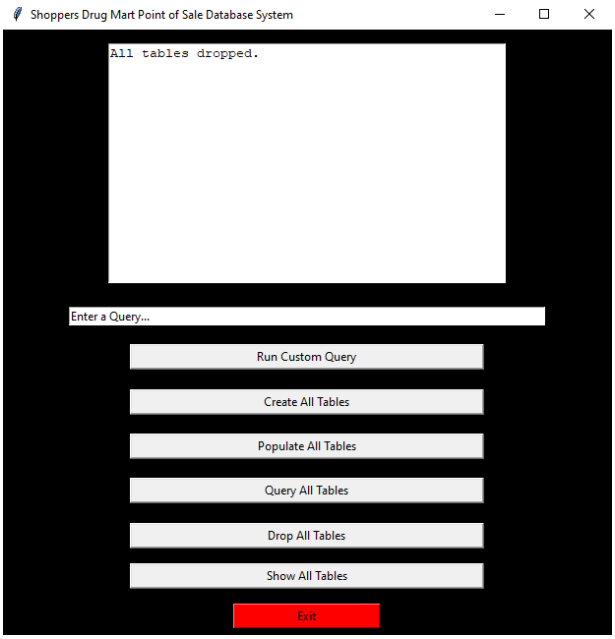
```
    try:
        #Run all the commands one by one
        for command in sql_commands:
            self.cursor.execute(command)
        self.connection.commit()
        self.result.insert(END, "All tables created.\n")
    #Log any errors incase tables already exist.
    except cx_Oracle.DatabaseError as e:
        self.result.insert(END, f"Error occured when creating the tables: {str(e)}\n")
    #Clear the field
    finally:
        self.result.config(state=DISABLED)

def query_all_tables(self):
    # Just some example queries from our pdf, more examples can be found
    queries = [
        "SELECT * FROM product WHERE price > 5",
        "SELECT * FROM customer",
        "SELECT payment_method AS \"Payment_Method\", SUM(total_price) AS \"Total Sales\" FROM receipt GROUP BY pa
        "SELECT name, 'Works as a ' || position || ' for Shoppers Drug Mart' AS \"Job Description\" FROM employee"
        "SELECT name || '''s Optimum ID is: ' || optimum_id AS \"Total points and Optimum IDs\" FROM customer UNIO
    ]
    #Clear previous text
    self.result.config(state=NORMAL)
    self.result.delete("1.0", END)
    #Execute each query in the list
    #Catch and log any errors to avoid program crashing
    for query in queries:
        try:
            #Run query
            self.cursor.execute(query)
            rows = self.cursor.fetchall()
            if rows:
                self.result.insert(END, f"Results for: {query}\n")
                for row in rows:
                    self.result.insert(END, f"{row}\n")
                #Format the output nicely
                self.result.insert(END, "\n" + "-"*50 + "\n\n")
            else:
                self.result.insert(END, f"No results for: {query}\n")
                self.result.insert(END, "\n" + "-"*50 + "\n\n")
        #Log any errors
        except Exception as e:
            self.result.insert(END, f"Error executing query: {query}\n")
            self.result.insert(END, f"Error message: {e}\n")

    self.result.config(state=DISABLED)

def drop_all_tables(self):
    #All of our drop table commands
    drop_table_commands = [
        "DROP TABLE receipt CASCADE CONSTRAINTS",
        "DROP TABLE transaction CASCADE CONSTRAINTS",
        "DROP TABLE customer CASCADE CONSTRAINTS",
        "DROP TABLE employee CASCADE CONSTRAINTS",
        "DROP TABLE inventory CASCADE CONSTRAINTS",
        "DROP TABLE product CASCADE CONSTRAINTS",
        "DROP TABLE optimum CASCADE CONSTRAINTS",
    ]
    #Clear previous text
    self.result.config(state=NORMAL)
    self.result.delete("1.0", END)
    self.result.insert(END, "All tables dropped.\n")
    #Drop the tables one by one
    for command in drop_table_commands:
        self.cursor.execute(command)
        self.connection.commit()

def display_all_tables(self):
    #Display all the tables currently in the DB
    self.result.config(state=NORMAL)
    self.result.delete("1.0", END)

    #Catch any errors incase no tables exist in DB
    try:
        self.cursor.execute("SELECT table_name FROM user_tables ORDER BY table_name")
        tables = self.cursor.fetchall()

        if tables:
            self.result.insert(END, "Current Tables in the Database:\n")
            for table in tables:
                self.result.insert(END, f"- {table[0]}\n")
        else:
            self.result.insert(END, "There are no tables currently in the database. Click the Create All Tables bu
    #Log error if any
    except cx_Oracle.DatabaseError as e:
        self.result.insert(END, f"Error retrieving tables: {str(e)}\n")
    finally:
        self.result.config(state=DISABLED)
```

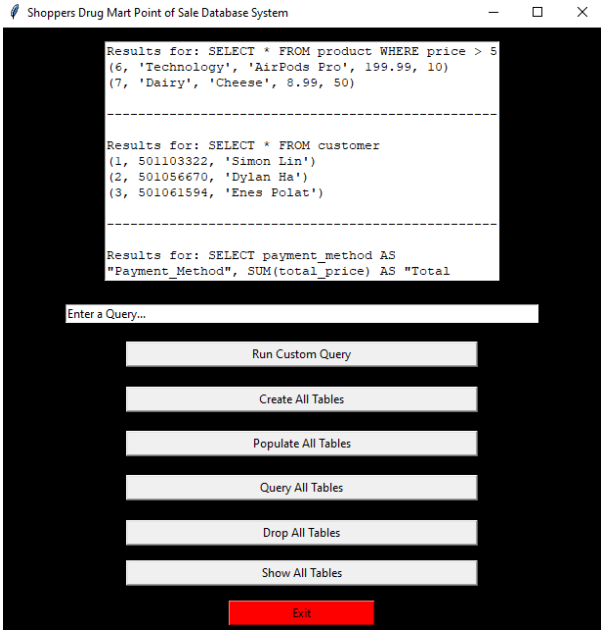
```
def populate_all_tables(self):
    #Populate all our tables with our dummy data.
    self.result.config(state=NORMAL)
    self.result.delete("1.0", END)
    sql_commands = [
        "INSERT INTO product VALUES(1, 'Fruit', 'Apple', 0.99, 10)",
        "INSERT INTO product VALUES(2, 'Beverage', 'Orange Juice', 2.99, 20)",
        "INSERT INTO product VALUES(3, 'Snack', 'Chips', 1.99, 15)",
        "INSERT INTO product VALUES(4, 'Dairy', 'Milk', 2.49, 25)",
        "INSERT INTO product VALUES(5, 'Vegetable', 'Potato', 1, 20)",
        "INSERT INTO product VALUES(6, 'Technology', 'AirPods Pro', 199.99, 10)",
        "INSERT INTO product VALUES(7, 'Dairy', 'Cheese', 8.99, 50)",

        "INSERT INTO inventory VALUES(1, 'Fruit', 'Apple',  50)",
        "INSERT INTO inventory VALUES(2, 'Beverage', 'Orange Juice',  50)",
        "INSERT INTO inventory VALUES(3, 'Snack', 'Chips',  50)",
        "INSERT INTO inventory VALUES(4, 'Dairy', 'Milk', 50)",

        "INSERT INTO optimum VALUES(501103322, 10000, 'Simon Lin')",
        "INSERT INTO optimum VALUES(501056670, 12000, 'Dylan Ha')",
        "INSERT INTO optimum VALUES(501061594, 8110, 'Enes Polat')",

        "INSERT INTO customer VALUES(1, 501103322, 'Simon Lin')",
        "INSERT INTO customer VALUES(2, 501056670, 'Dylan Ha')",
        "INSERT INTO customer VALUES(3, 501061594, 'Enes Polat')",

        "INSERT INTO employee VALUES(3, 'Cashier', 'Ski Betty')",
        "INSERT INTO employee VALUES(2, 'Manager', 'Hawk T. Ooah')",
        "INSERT INTO employee VALUES(1, 'Owner', 'Hugh Mungus')",

        "INSERT INTO transaction VALUES(1, 3, 500, 23.59, 'Cash', CURRENT_DATE)",
        "INSERT INTO transaction VALUES(2, 2, 300, 20, 'Debit', CURRENT_DATE)",
        "INSERT INTO transaction VALUES(3, 3, 600, 60, 'Credit', CURRENT_DATE)",
        "INSERT INTO transaction VALUES(4, 1, 100, 20.23, 'Credit', CURRENT_DATE)",
        "INSERT INTO transaction VALUES(5, 3, 200, 2, 'Debit', CURRENT_DATE)",
        "INSERT INTO transaction VALUES(6, 2, 200, 50, 'Cash', CURRENT_DATE)",
        "INSERT INTO transaction VALUES(7, 1, 300, 20, 'Cash', CURRENT_DATE)",
        "INSERT INTO transaction VALUES(8, 1, 300, 50, 'Cash', CURRENT_DATE)",

        "INSERT INTO receipt VALUES(1, '22 Apples', 500, 23.59, 'Cash', CURRENT_DATE)",
        "INSERT INTO receipt VALUES(2, '10 Chips', 300, 20.00, 'Debit', CURRENT_DATE)",
        "INSERT INTO receipt VALUES(3, '15 Apples', 600, 60.00, 'Credit', CURRENT_DATE)",
        "INSERT INTO receipt VALUES(4, '8 Milk', 100, 20.23, 'Credit', CURRENT_DATE)",
        "INSERT INTO receipt VALUES(5, '2 Potatoes', 200, 2.00, 'Debit', CURRENT_DATE)",
        "INSERT INTO receipt VALUES(6, '16 Orange Juice', 200, 50.00, 'Cash', CURRENT_DATE)",
        "INSERT INTO receipt VALUES(7, '20 Apples', 300, 20.00, 'Cash', CURRENT_DATE)",
        "INSERT INTO receipt VALUES(8, '50 Apples', 300, 50.00, 'Cash', CURRENT_DATE)"
    ]
    #Catch any errors incase no tables exist in DB or faulty data.
    try:
        for command in sql_commands:
            self.cursor.execute(command)
        self.connection.commit()
        self.result.insert(END, "All tables populated.\n")
    except cx_Oracle.DatabaseError as e:
        self.result.insert(END, f"Error occured when populating tables: {str(e)}\n")
    finally:
        self.result.config(state=DISABLED)

#Display frame
def display(self, frame):
    frame.tkraise()

#Exit the UI
def exit(self):
    if self.cursor:
        self.cursor.close()
    if self.connection:
        self.connection.close()
    self.root.quit()

#Close the window.
def close_window(self):
        self.root.quit()
        self.root.destroy()

# for our custom query field
def field(self):
        if self.input.get() == "Enter a Query...":
            self.input.delete(0, END)
```

main.py

```python
from tkinter import *
from PIL import ImageTk, Image
from dbfunctions import custom_query, create_all_tables, query_all_tables, populate_all_tables, exit, field, close_window, drop_all_tables, display_all_t
from dbconnect import connection

class POSShoppers:
    def __init__(self, root):
        #Declare window attributes and title
        self.root = root
        self.root.title("Shoppers Drug Mart Point of Sale Database System")
        self.root.geometry("620x620")

        self.login = Frame(root)
        self.functions = Frame(root)
        #Declare oracle variables
        self.connection = None
        self.cursor = None

        #Change background color
        self.functions = Frame(root, bg="black")
        for frame in (self.login, self.functions):
            frame.grid(row=0, column=0, sticky="news")

        root.grid_rowconfigure(0, weight=1)
        root.grid_columnconfigure(0, weight=1)

        #Initialize functions
        self.init_login_page()
        self.init_functions()

        #Show the UI
        display(self, self.login)

        self.root.protocol("WM_DELETE_WINDOW", lambda: close_window(self))

    def init_login_page(self):
        for i in range(5):
            self.login.grid_rowconfigure(i, weight=1)
        self.login.grid_columnconfigure(0, weight=1)

        #Display shoppers logo
        image = Image.open("POSShoppers_CPS510A9/logo.png")
        image = image.resize((300, 250))
        img = ImageTk.PhotoImage(image)

        #Create login field
        label = Label(self.login, image=img)
        label.image = img
        label.grid(row=0, column=0, pady=5)

        self.login_message = Label(self.login, text="Enter CS Username and Password to Login")
        self.login_message.grid(row=1, column=0, pady=5)

        #Create the username and password field
        username = Frame(self.login)
        Label(username, text="Username").pack(side=LEFT, padx=5)
        self.username_input = Entry(username, width=30)
        self.username_input.pack(side=RIGHT, padx=5)
        username.grid(row=2, column=0, pady=1)

        password = Frame(self.login)
        Label(password, text="Password").pack(side=LEFT, padx=5)
        self.password_input = Entry(password, show="*", width=30) #
        self.password_input.pack(side=RIGHT, padx=5)
        password.grid(row=3, column=0, pady=5)

        login = Button(self.login, text="Login", command=lambda:connection(self))
        login.grid(row=4, column=0, pady=10)

    #Initialize all the buttons and their functions
    def init_functions(self):
        for i in range(7):
            self.functions.grid_rowconfigure(i, weight=1)
        self.functions.grid_columnconfigure(0, weight=1)

        self.result = Text(self.functions, wrap=WORD, state=DISABLED, height=15, width=50)
        self.result.grid(row=0, column=0, padx=10, pady=10)

        self.input = Entry(self.functions, width=80) #Create an input field
        self.input.insert(0, "Enter a Query...")
        self.input.bind("<FocusIn>", lambda _: field(self))
        self.input.grid(row=1, column=0, padx=10, pady=5)

        #Buttons and attribute declaration
        Button(self.functions, text="Run Custom Query", width=50, command=lambda: custom_query(self)).grid(row=2, column=0, pady=5)
        Button(self.functions, text="Create All Tables", width=50, command=lambda: create_all_tables(self)).grid(row=3, column=0, pady=5)
        Button(self.functions, text="Populate All Tables", width=50, command=lambda: populate_all_tables(self)).grid(row=4, column=0, pady=5)
        Button(self.functions, text="Query All Tables", width=50, command=lambda: query_all_tables(self)).grid(row=5, column=0, pady=5)
        Button(self.functions, text="Drop All Tables", width=50, command=lambda: drop_all_tables(self)).grid(row=6, column=0, pady=5)
        Button(self.functions, text="Show All Tables", width=50, command=lambda: display_all_tables(self)).grid(row=7, column=0, pady=5)
        Button(self.functions, text="Exit", width=20, bg='red', command=lambda: exit(self)).grid(row=8, column=0, pady=10)

if __name__ == "__main__":
    root = Tk()
    app = POSShoppers(root)
    root.mainloop()
```


Shoppers Drug Mart Point of Sale Database System
Current Tables in the Database:
- CUSTOMER
- EMPLOYEE
- INVENTORY
- OPTIMUM
- PRODUCT
- RECEIPT
- TRANSACTION
Enter a Query...
Run Custom Query
Create All Tables
Populate All Tables
Query All Tables
Drop All Tables
Show All Tables
Exit


Shoppers Drug Mart Point of Sale Database System
All tables dropped.
Enter a Query...
Run Custom Query
Create All Tables
Populate All Tables
Query All Tables
Drop All Tables
Show All Tables
Exit


Shoppers Drug Mart Point of Sale Database System
Results for: SELECT * FROM product WHERE price > 5
(6, 'Technology', 'AirPods Pro', 199.99, 10)
(7, 'Dairy', 'Cheese', 8.99, 50)
Results for: SELECT * FROM customer
(1, 501103322, 'Simon Lin')
(2, 501056670, 'Dylan Ha')
(3, 501061594, 'Enes Polat')
Results for: SELECT payment_method AS
"Payment_Method", SUM(total_price) AS "Total
Enter a Query...
Run Custom Query
Create All Tables
Populate All Tables
Query All Tables
Drop All Tables
Show All Tables
Exit


Shoppers Drug Mart Point of Sale Database System
(1, 501103322, 'Simon Lin')
(2, 501056670, 'Dylan Ha')
(3, 501061594, 'Enes Polat')
select * from customer
Run Custom Query
Create All Tables
Populate All Tables
Query All Tables
Drop All Tables
Show All Tables
Exit