Assignment 2

Tutor: Dr Oliver Buckley

Submission date: 25 October 2021

Question: System Implementation

The deliverables for this assignment are:

1. Python code with comments explaining the code
2. README file containing a description of the solution implemented and instructions on how to execute the code
3. Test data used to test the code

Table of Contents

1. Introduction

The second part of this assignment is on creating/implementing the system in Python, by using the models created in the first part of the assignment which was on designing and developing some UML diagrams (class diagram, activity diagram, state diagram, sequence diagram) for an online shopping centre.

The following class diagram and activity diagram which were created in the first part of the assignment will be partly used as models during the implementation in Python:

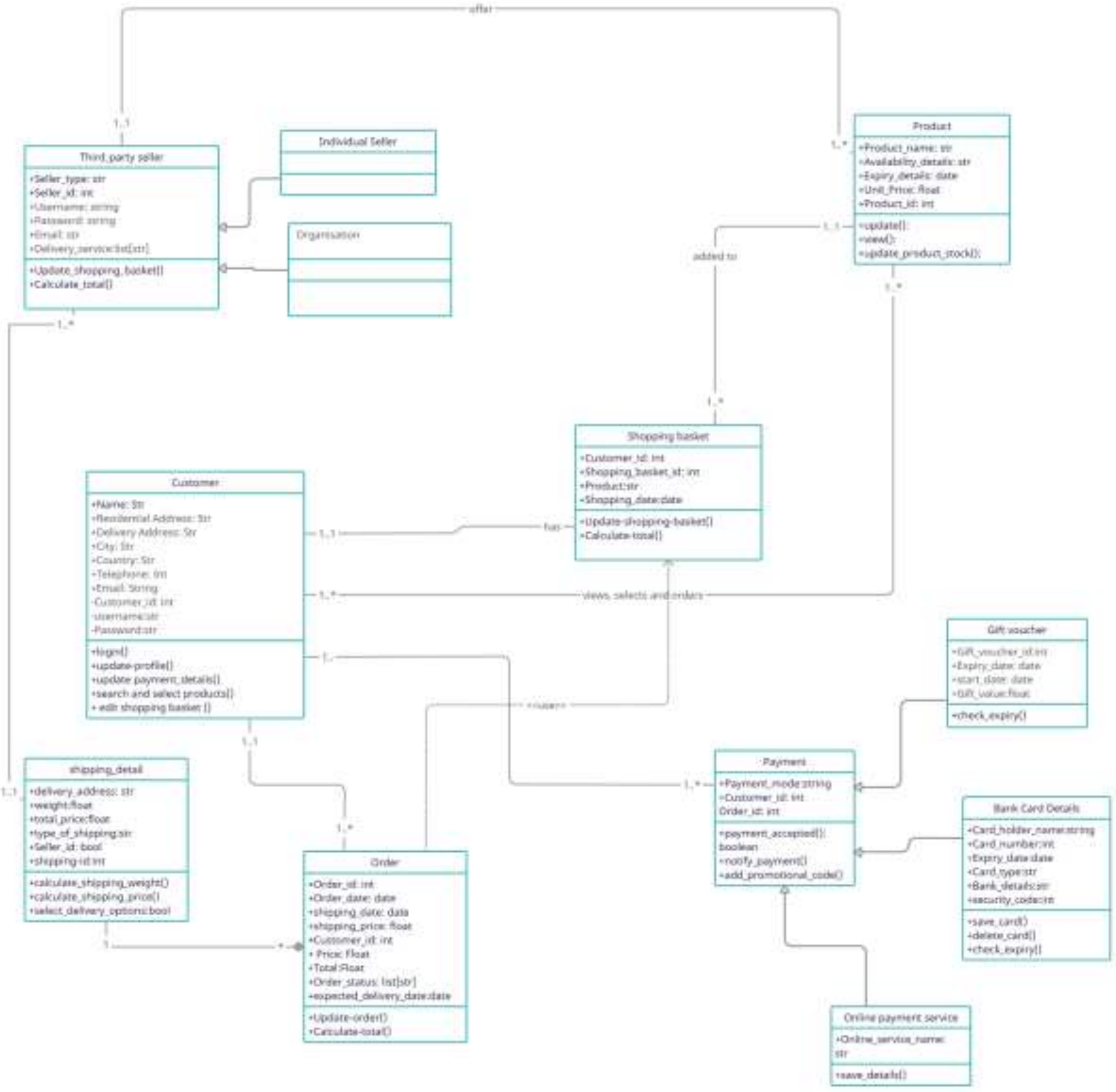


Figure 1:Class Diagram (from Assignment 1)

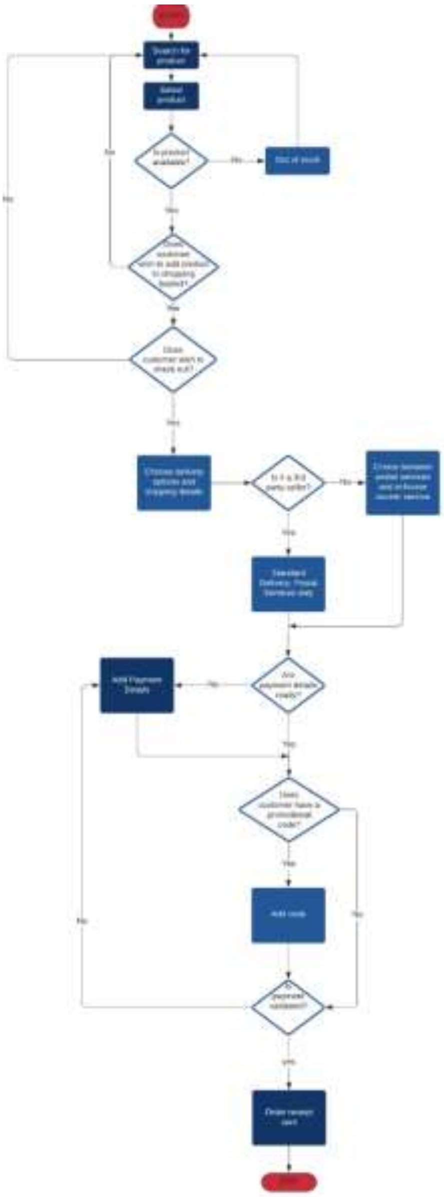

Search for product
Select product
Is product available?
No
Out of stock
Yes
No
Does customer wish to add product to shopping basket?
Yes
No
Does customer wish to check out?
Yes
Choose delivery options and shipping details
Is it a 3rd party seller?
No
Choose between postal services and in-house courier service
Yes
Standard Delivery: Postal Services only
Are payment details ready?
No
Add Payment Details
Yes
Does customer have a promotional code?
Yes
Add code
No
No
Is payment validated?
yes
Order receipt sent

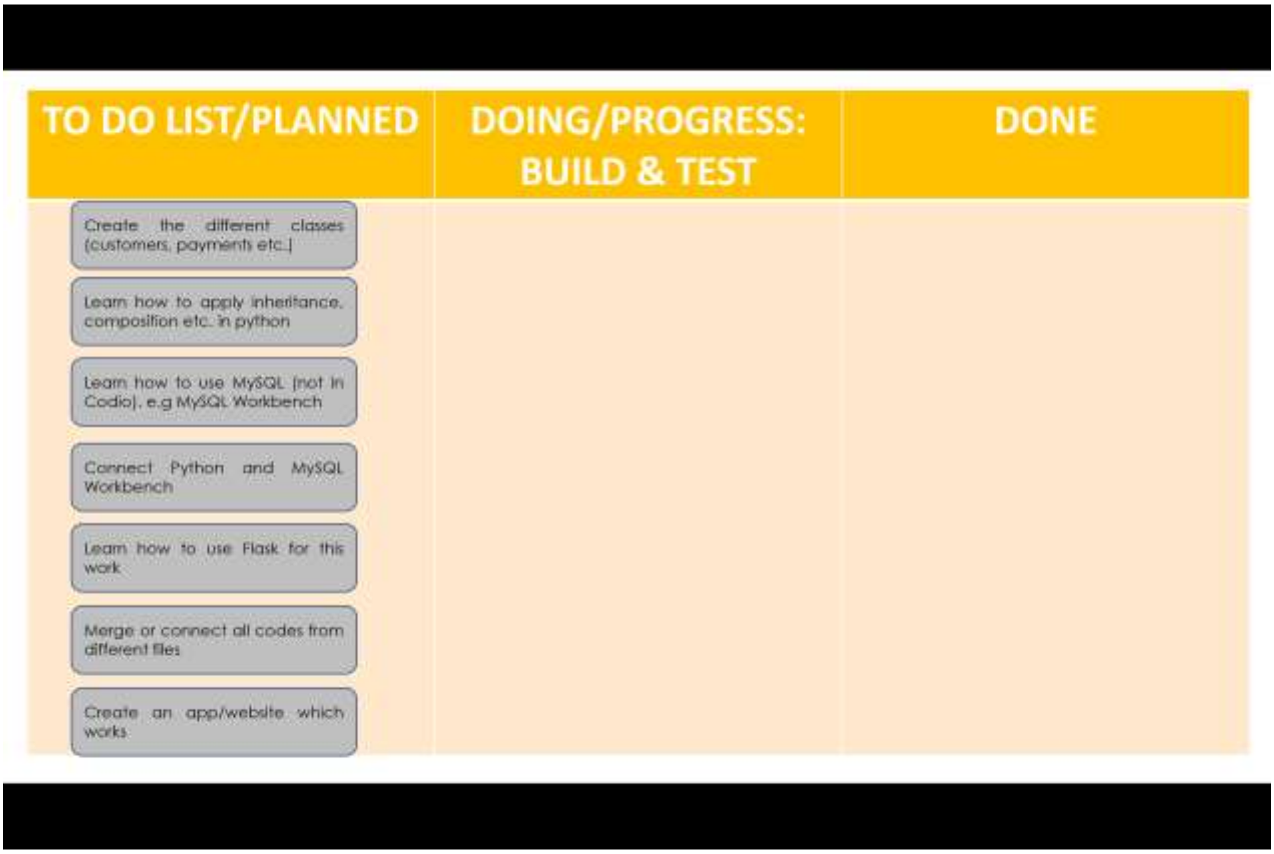
Figure 2: Activity Diagram

The tutor’s comments were quite positive overall, and one of the main improvements to be brought was with the “use of composition and inheritance”: “there are several occasions where composition would have been useful, for example, an order containing a customer” (Buckley, Feedback on Assignment 1, 2021).

Visual Studio Code and IDLE (Python) were used to create this system. Instead of using SQL in Codio, University of Essex Online's platform/IDE, mysql workbench was installed and used by following online tutorials (Programming with Mosh, 20 March 2019; Codebasics, 16 November 2020). This would be helpful for future personal and professional projects.

A root/main project directory was set up in VS Code, and connections made with a launch.json file to use Python.

The Kanban chart below is a tentative workplan to prioritise tasks for the creation of the Online Shopping Centre Information System:



TO DO LIST/PLANNED	DOING/PROGRESS: BUILD & TEST	DONE
Create the different classes (customers, payments etc.)		
Learn how to apply inheritance, composition etc. in python		
Learn how to use MySQL (not in Codio), e.g MySQL Workbench		
Connect Python and MySQL Workbench		
Learn how to use Flask for this work		
Merge or connect all codes from different files		
Create an app/website which works		

Figure 3: To Do List

2. README FILE

(Brief commentary on the solution implemented, in relation to the design discussed in assignment 1)

2.1. TITLE

Online Shopping Centre Information System (OSCIS)

2.2. DESCRIPTION

This Online Shopping Centre Information System (OSCIS), is an open-source application/project, designed for an online/e-commerce.

The main attractivity of this OSCIS is that users can register themselves and input their required information directly in the system. They can search, select, cancel or pay for products online. The app/site administrator will also have access to add/delete/change/sort information more easily.

2.3. INSTRUCTIONS TO RUN THE APPLICATION

The application is built with Python and MySQL is connected to it. Flask, as a lightweight Python Framework for creating web applications, was used to display and order the products.

2.4. THE FEATURES

Our first aim was to create our classes and tables in MYSQL. Python and MySQL were connected. Flask was also used. A detailed explanation is given in the section below.

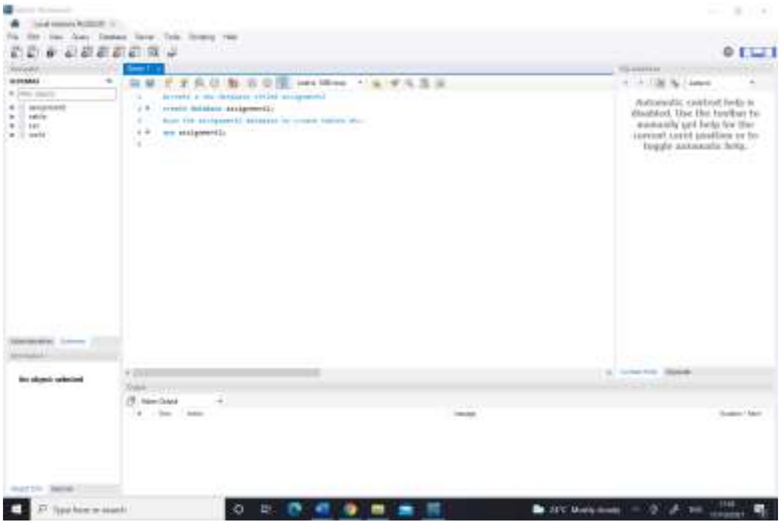
3. Detailed Explanatory Note on this Project Implementation

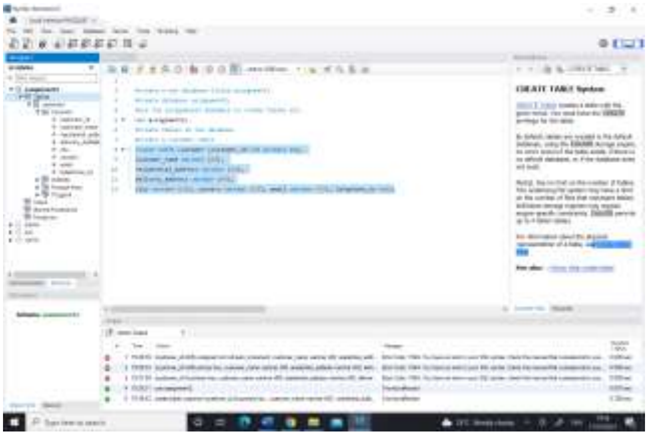
3.1. MYSQL

Two methods were learnt to work with MYSQL and Python.

3.1.1. USING MYSQL WORKBENCH AND PYTHON

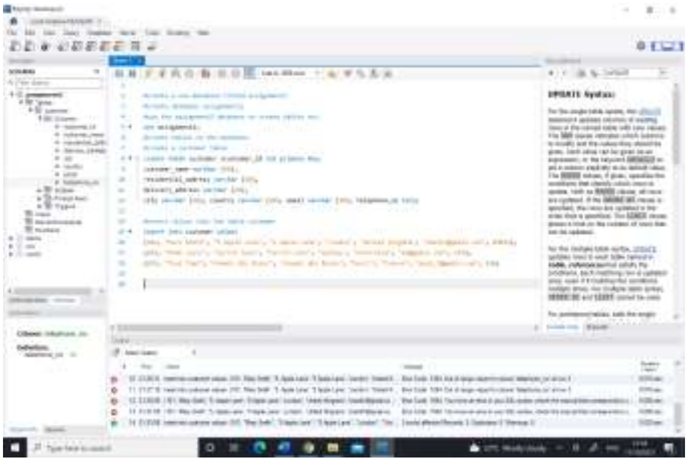
As shown in the figures below, a database “Assignment 2” was created in MySQL Workbench.





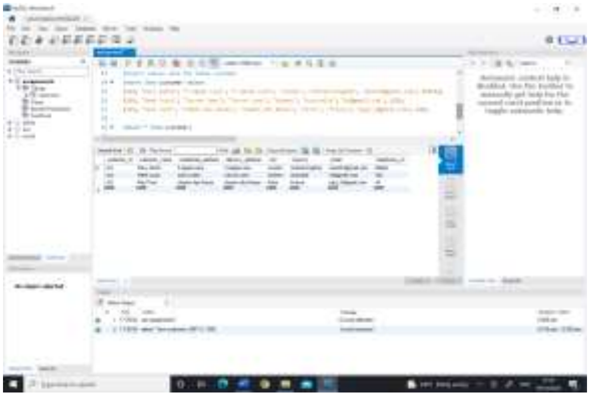
Remember to run “use the database” you want first every time… otherwise table won’t be created

Values were inserted in order to test if the queries are being executed properly:

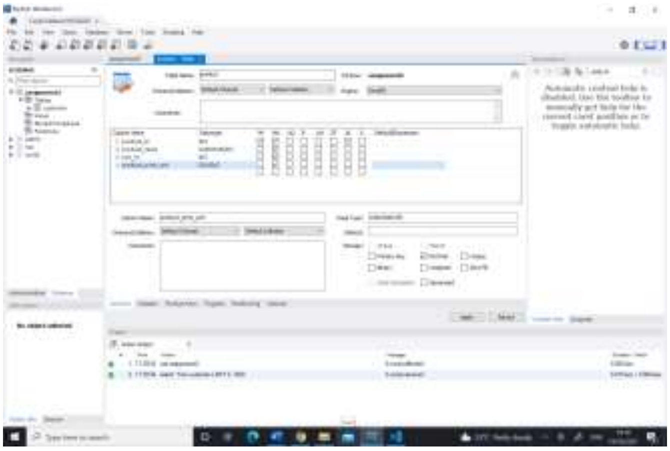


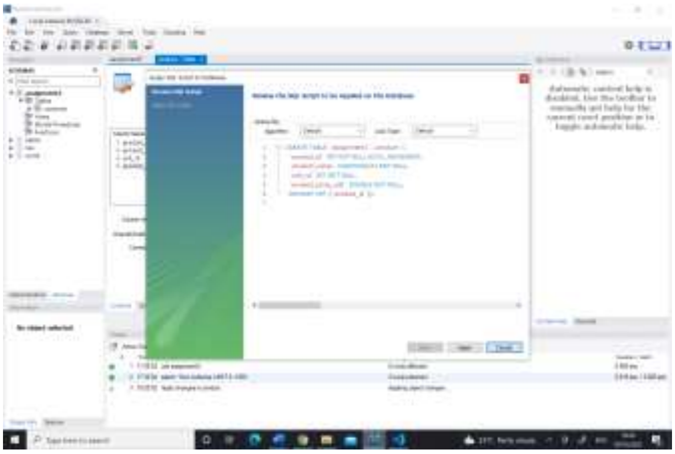
Insert double inverted commas and not single inverted commas

The table with the requested values were created after having rectified the single/double inverted commas issues:



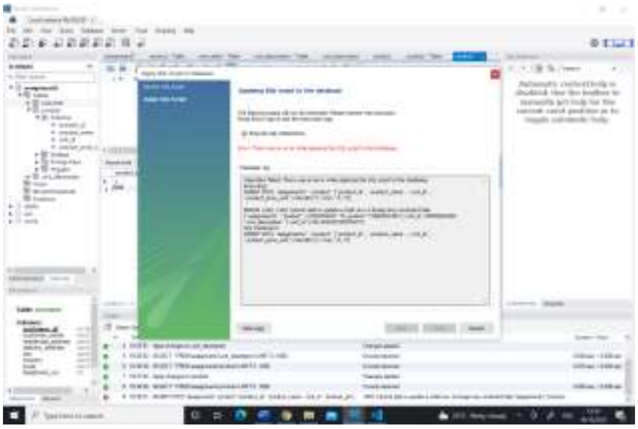
Tables can also be automatically created in mySQL Workbench as shown in the tables below. The product_table was created in this way (Codebasics, 17 November 2020):

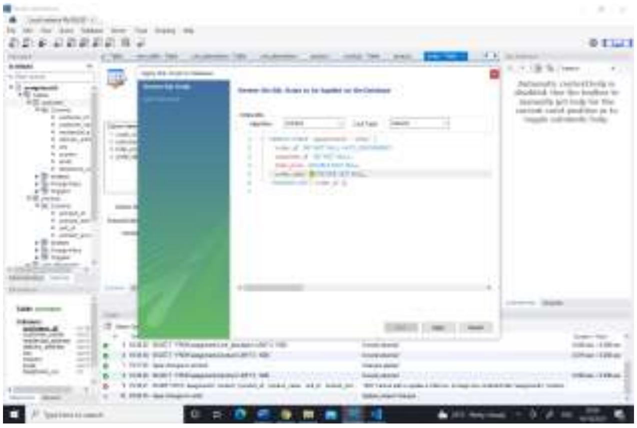




A unit_description table was created to link each product with its unit (kg or each).

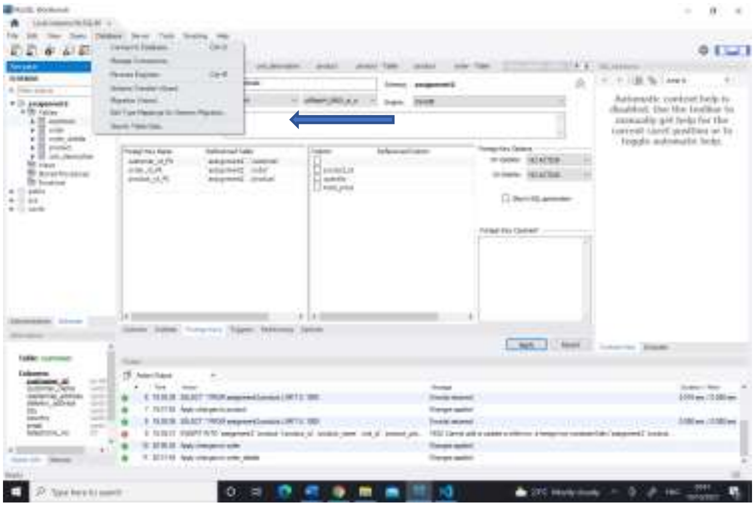
Testing was done to check if foreign key works.

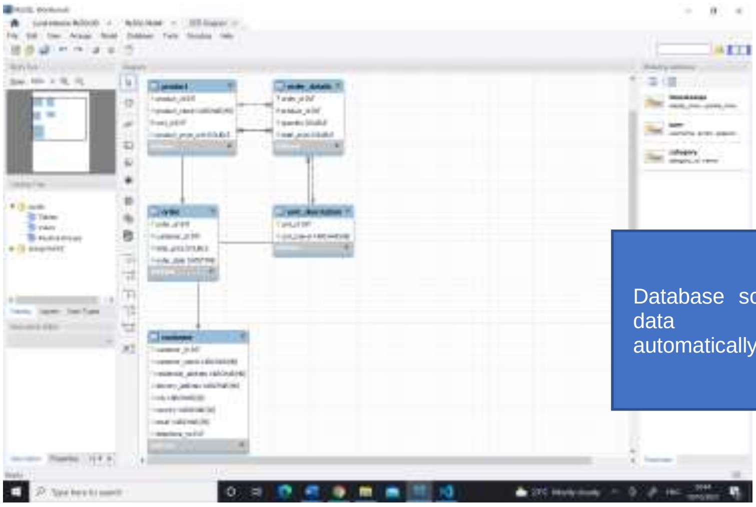




ERROR in datetime(). Remove brackets

A relation diagram can be created directly in MySQL workbench:





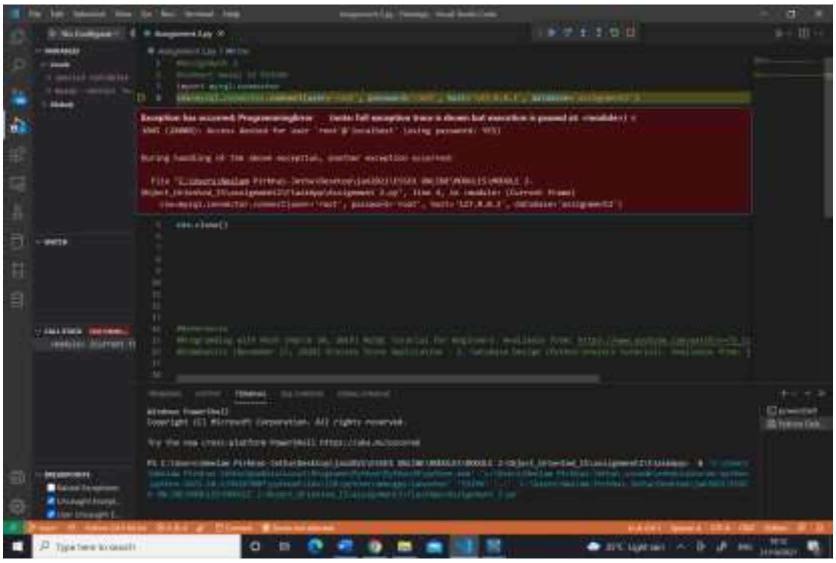
Database schema or data model automatically created.

3.1.2. Connecting MySQL Workbench to Python

3.1.2.1. Connecting…

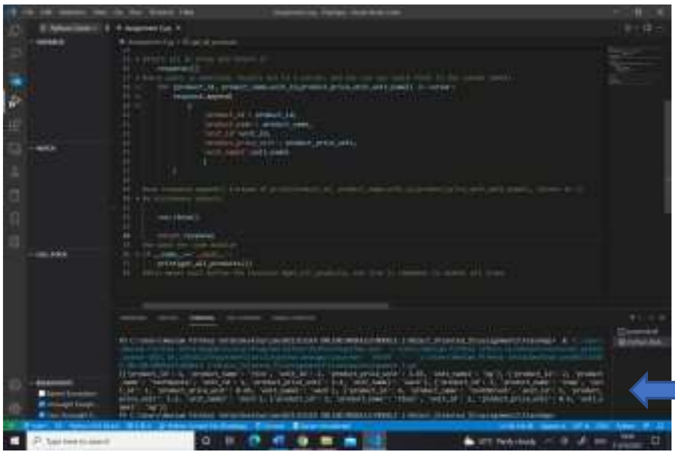
[Kindly check submitted file: products_dao.py (renamed) and sql_connection.py for codes]

There was an error while typing the connection code, but after having copied and pasted the code from MYSQL documentation (Oracle, 2021), it seems to work.

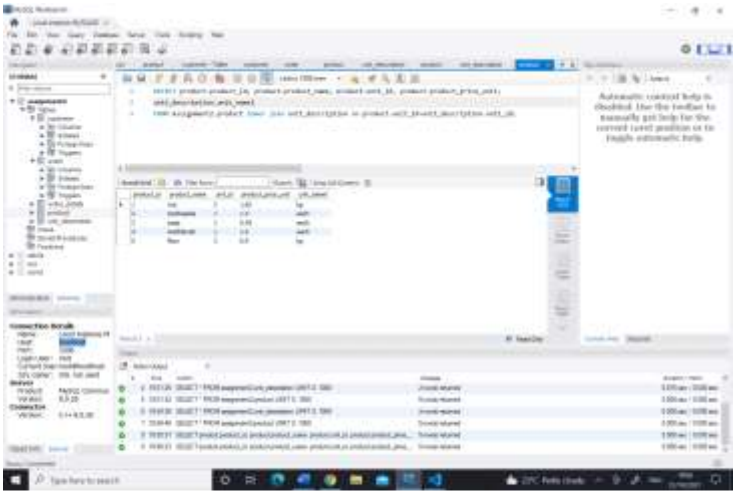


```
import mysql.connector
[illegible]
Exception has occurred: ProgrammingError       (note: full exception trace is shown but execution is paused at: <module>)
1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

During handling of the above exception, another exception occurred:

  File [illegible], line 4, in <module> (Current frame)
    [illegible]
cnx.close()
```

The list of products appears when the programme is executed as shown below:



However, kg etc doesn’t appear. The tables product and unit_description created in MySQL must be joined. See inner join below

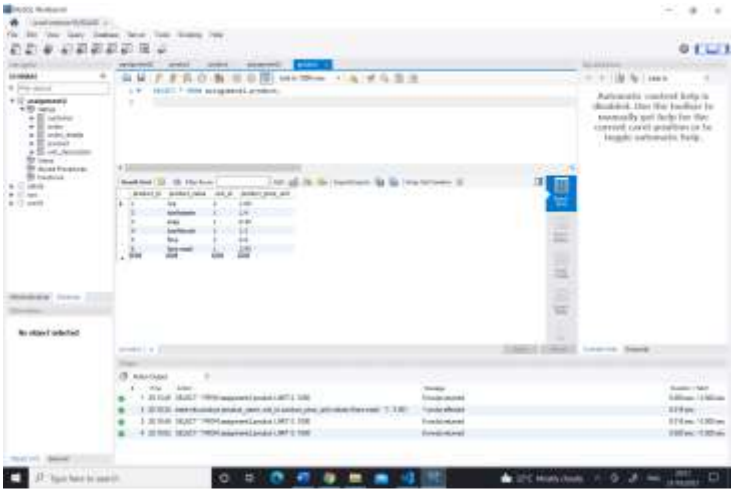


There are two ways of importing MYSQL to python and creating the classes among others. The first one is to put all codes in one python file. The second way is to make the code more modular by creating a connection with SQL in another python file (codebasics, 2020). The following submitted python files are connected:

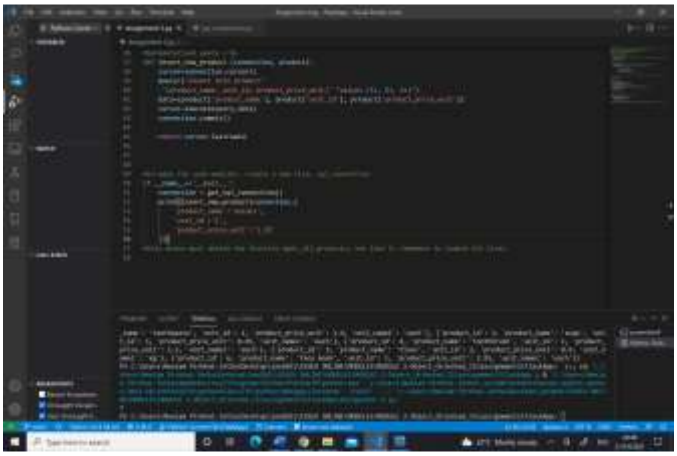
- products.dao.py
- sql.connections.py

3.1.2.2. Inserting and Deleting Products in both MySQL and Python

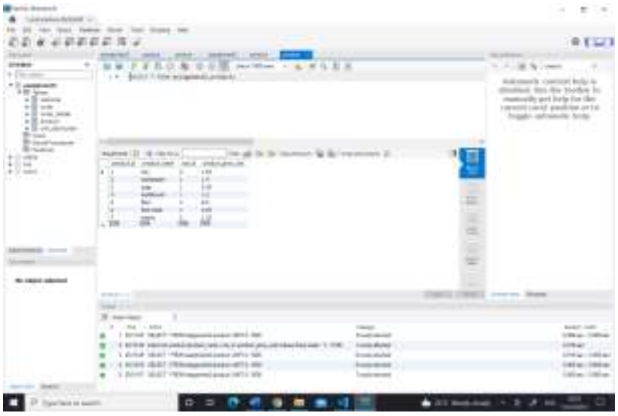
In the product file (SQL and Python), the functions to insert product and delete product were defined. For instance, the product “Onions” doesn’t exist yet, and has been created as shown below:



No product ‘onion’

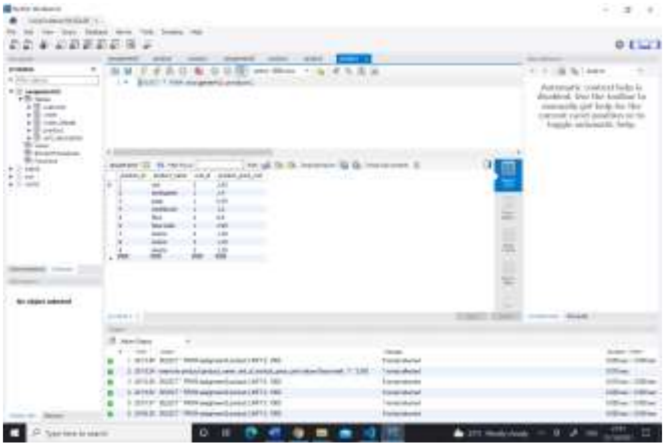


Codes in python (connected to database)

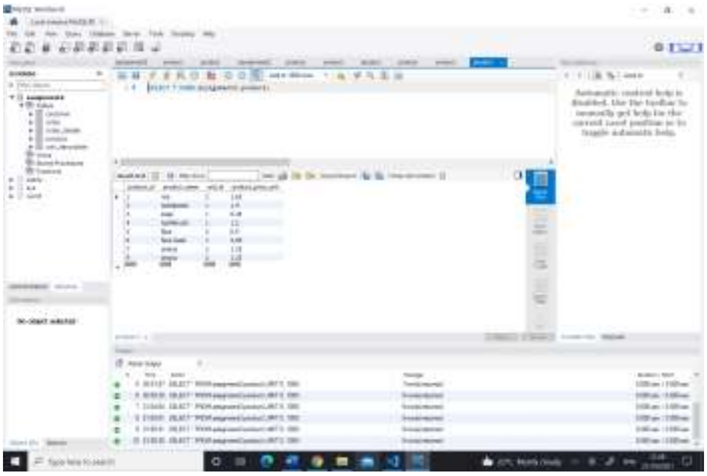


Product ‘onion’ automatically inserted in MYSQL

The following codes were used to remove query 8 and 9, for instance: onions are repeated



Query 8 removed.



3.1.3. The 2nd Method: Using SQLite3

Install <pip install flask-sqlalchemy' in the command prompt. The package will make it possible to work with a database in Python. Code to import flask-SQLAlchemy is available in the market.py file.

```
#import sqlite3
from flask_sqlalchemy import SQLAlchemy
#instantiate database (convention: use db)
db=SQLAlchemy(app)
```

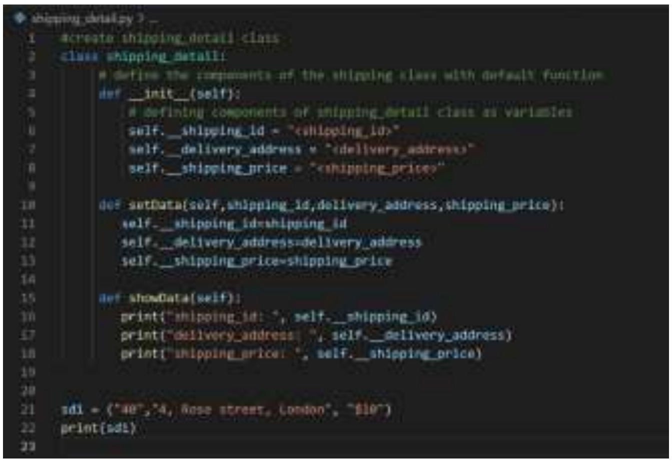
Classes can be created and these will be converted into a database tables (known as Model)

A few issues were met when typing in the commands in the command line – could not import db in the python file (market.py). This method will be learnt later.

3.2. Creating Classes in Python

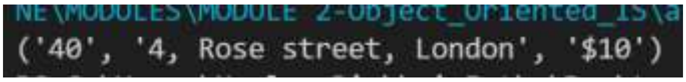
3.2.1. Codes for a class

Kindly check shipping_detail.py for codes.



shipping_detail.py > ...

```python
#create shipping_detail class
class shipping_detail:
        # define the components of the shipping class with default function
        def __init__(self):
            # defining components of shipping_detail class as variables
            self.__shipping_id = "<shipping_id>"
            self.__delivery_address = "<delivery_address>"
            self.__shipping_price = "<shipping_price>"

        def setData(self,shipping_id,delivery_address,shipping_price):
           self.__shipping_id=shipping_id
           self.__delivery_address=delivery_address
           self.__shipping_price=shipping_price

        def showData(self):
           print("shipping_id: ", self.__shipping_id)
           print("delivery_address: ", self.__delivery_address)
           print("shipping_price: ", self.__shipping_price)


sd1 = ("40","4, Rose street, London", "$10")
print(sd1)
```

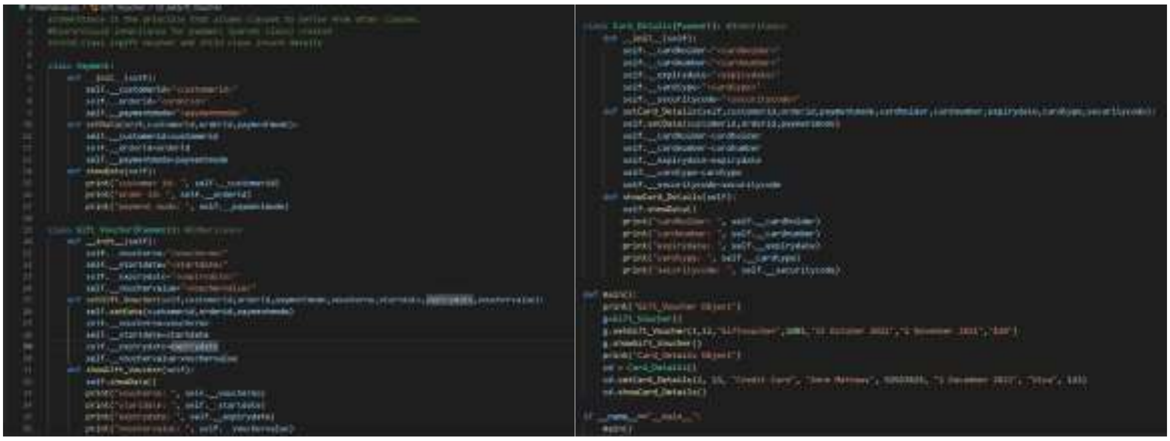


```
NE\MODULES\MODULE 2-Object_Oriented_IS\a
('40', '4, Rose street, London', '$10')
```

3.2.2. Inheritance

[Kindly check submitted file: inheritance.py for codes]

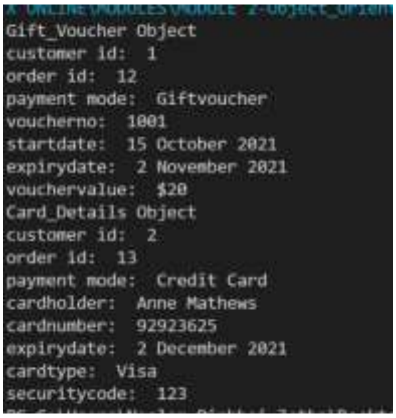
The superclass/parent class “Payment” was created. Two subclasses were also created to show the two modes of payment: Gift voucher and Card_Details. To understand inheritance codes, some sites were consulted (Buckley, 2021;Yeoman, 2021).



```python
#inheritance is the principle that allows classes to derive from other classes.
#superclass(parent class) for payment [illegible]
#child class (gift voucher and child class (card details)

class Payment:
    def __init__(self):
        self.__customerid="customerid"
        self.__orderid="orderid"
        self.__paymentmode="paymentmode"
    def setData(self,customerid,orderid,paymentmode):
        self.__customerid=customerid
        self.__orderid=orderid
        self.__paymentmode=paymentmode
    def showData(self):
        print("customer ID: ", self.__customerid)
        print("order ID: ", self.__orderid)
        print("payment mode: ", self.__paymentmode)

class Gift_Voucher(Payment): #inheritance
    def __init__(self):
        self.__vouchername="vouchername"
        self.__startdate="startdate"
        self.__expirydate="expirydate"
        self.__vouchervalue="vouchervalue"
    def setGift_Voucher(self,customerid,orderid,paymentmode,vouchername,startdate,expirydate,vouchervalue):
        self.setData(customerid,orderid,paymentmode)
        self.__vouchername=vouchername
        self.__startdate=startdate
        self.__expirydate=expirydate
        self.__vouchervalue=vouchervalue
    def showGift_Voucher(self):
        self.showData()
        print("vouchername: ", self.__vouchername)
        print("startdate: ", self.__startdate)
        print("expirydate: ", self.__expirydate)
        print("vouchervalue: ", self.__vouchervalue)
```

```python
class Card_Details(Payment): #inheritance
    def __init__(self):
        self.__cardholder="cardholder"
        self.__cardnumber="cardnumber"
        self.__expirydate="expirydate"
        self.__cardtype="cardtype"
        self.__securitycode="securitycode"
    def setCard_Details(self,customerid,orderid,paymentmode,cardholder,cardnumber,expirydate,cardtype,securitycode):
        self.setData(customerid,orderid,paymentmode)
        self.__cardholder=cardholder
        self.__cardnumber=cardnumber
        self.__expirydate=expirydate
        self.__cardtype=cardtype
        self.__securitycode=securitycode
    def showCard_Details(self):
        self.showData()
        print("cardholder: ", self.__cardholder)
        print("cardnumber: ", self.__cardnumber)
        print("expirydate: ", self.__expirydate)
        print("cardtype: ", self.__cardtype)
        print("securitycode: ", self.__securitycode)

def main():
    print("Gift_Voucher Object")
    g=Gift_Voucher()
    g.setGift_Voucher(1,12,"GiftVoucher",[illegible],"[illegible] October 2021","1 November 2021","[illegible]")
    g.showGift_Voucher()
    print("Card_Details Object")
    cd = Card_Details()
    cd.setCard_Details(1, 15, "Credit Card", "[illegible]", [illegible], "1 December [illegible]", "Visa", [illegible])
    cd.showCard_Details()

if __name__=="__main__":
    main()
```

The codes function as shown below:



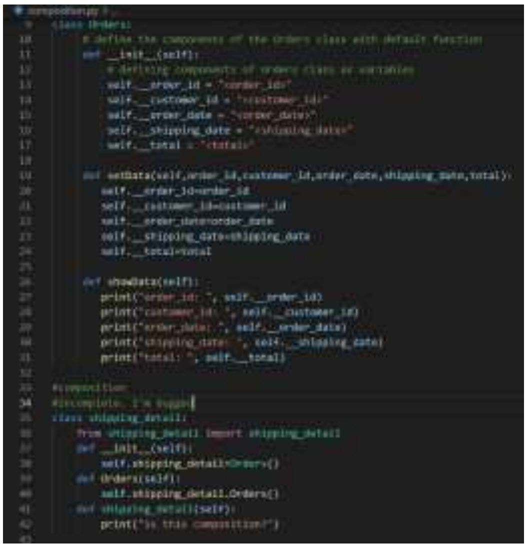
```
X ONLINE\MODULES\MODULE 2-Object_Orien
Gift_Voucher Object
customer id:  1
order id:  12
payment mode:  Giftvoucher
voucherno:  1001
startdate:  15 October 2021
expirydate:  2 November 2021
vouchervalue:  $20
Card_Details Object
customer id:  2
order id:  13
payment mode:  Credit Card
cardholder:  Anne Mathews
cardnumber:  92923625
expirydate:  2 December 2021
cardtype:  Visa
securitycode:  123
```

Check this site for inheritance and compositionhttps://www.youtube.com/watch?v=0mcP8ZpUR38

3.2.3. Composition

[Kindly check submitted file: composition.py for codes]

Composition is the act of collecting several objects together to create a new one. Composition is usually a good choice when one object is part of another object (Seidl et al, 2012). In our class diagram, a composition line is drawn from the class “shipping details” to the class “order”. If the “order” class is cancelled, then, the shipping details class is automatically cancelled. However, there were some issues with this part: it’s not working and must be checked again.



```
composition.py
class Orders:
    # define the components of the Orders class with default function
    def __init__(self):
        # defining components of orders class as variables
        self.__order_id = "<order_id>"
        self.__customer_id = "<customer_id>"
        self.__order_date = "<order_date>"
        self.__shipping_date = "<shipping_date>"
        self.__total = "<total>"

    def setData(self,order_id,customer_id,order_date,shipping_date,total):
        self.__order_id=order_id
        self.__customer_id=customer_id
        self.__order_date=order_date
        self.__shipping_date=shipping_date
        self.__total=total

    def showData(self):
        print("order_id: ", self.__order_id)
        print("customer_id: ", self.__customer_id)
        print("order_date: ", self.__order_date)
        print("shipping_date: ", self.__shipping_date)
        print("total: ", self.__total)

#composition
#incomplete. I'm buggin
class shipping_detail:
    from shipping_detail import shipping_detail
    def __init__(self):
        self.shipping_detail=Orders()
    def Orders(self):
        self.shipping_detail.Orders()
    def shipping_detail(self):
        print("is this composition?")
```

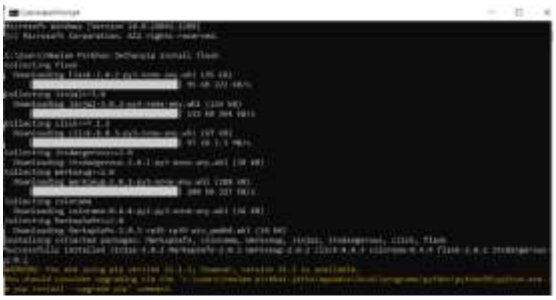
Explanation was found online (https://www.tutorialandexample.com/composition-in-python/), adapted and tested but a few steps were not clear and did not function.

3.3. Flask

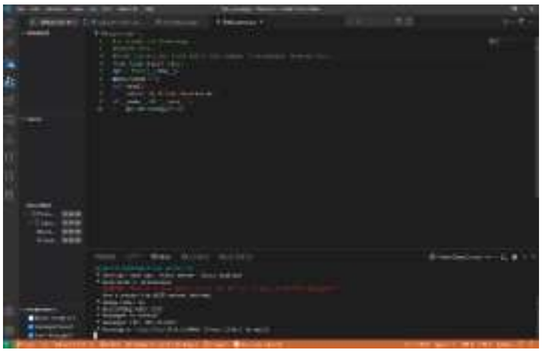
[Kindly check submitted files: Flask_server.py for the codes; homepage.html, market.html were the pages created]

3.3.1. Installing Flask

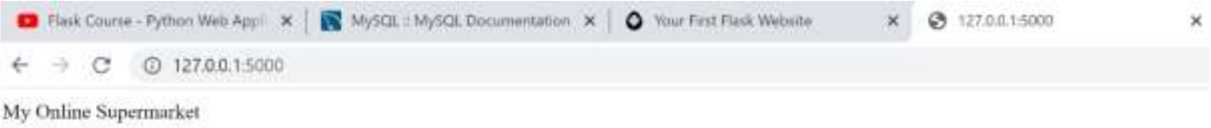
Flask was installed using the command prompt (JP Infotech Projects, 3 December 2020).



```
[illegible]
(c) Microsoft Corporation. All rights reserved.

[illegible] install flask
Collecting flask
[illegible]
Collecting itsdangerous[illegible]
[illegible]
Collecting colorama
[illegible]
Installing collected packages: MarkupSafe, colorama, Werkzeug, Jinja2, itsdangerous, click, flask
Successfully installed [illegible]
WARNING: You are using pip version [illegible]
You should consider upgrading via the [illegible]
```

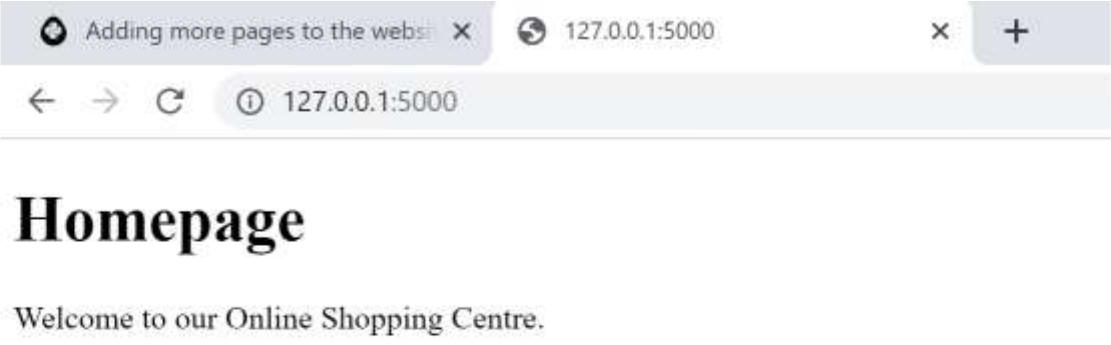


To access the homepage, after having run it in Python, click on: http://127.0.0.1:5000/
To access the market page, click on http://127.0.0.1:5000/flask_server



My Online Supermarket

HTML Templated added, and CSS can be used later…

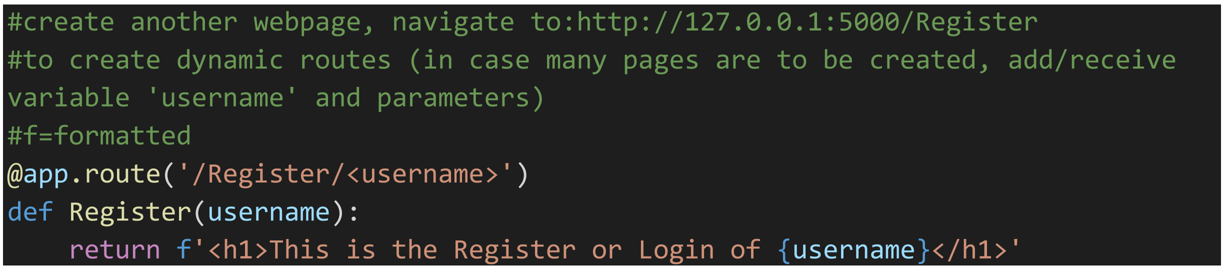


Homepage

Welcome to our Online Shopping Centre.

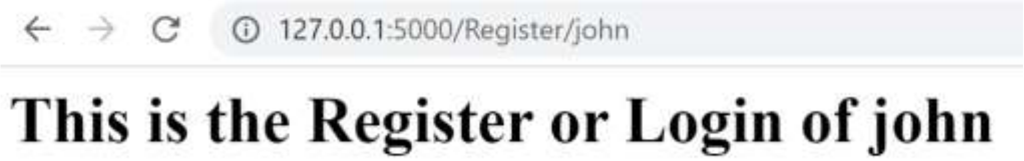
From the other video tutorial (freecodecamp.org) which was watched, a final python file was created (**renamed flask_server.py**).

Various pages can be created on Flask. This is known as dynamic routes, and the following codes must be implemented to make it possible:



```
#create another webpage, navigate to:http://127.0.0.1:5000/Register
#to create dynamic routes (in case many pages are to be created, add/receive variable 'username' and parameters)
#f=formatted
@app.route('/Register/<username>')
def Register(username):
    return f'<h1>This is the Register or Login of {username}</h1>'
```

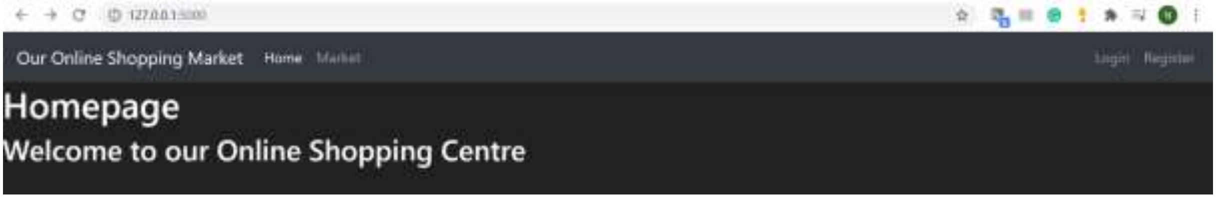
Output:



This is the Register or Login of john

However, to avoid too many codes and lines, instead of having all the pages (HTML codes) in one python file, it’s better to have a folder known as templates (template is a convention in python). Two pages were created: homepage and market.

A few additions to the page: navigation bar added, background colour etc. added:



Homepage

Welcome to our Online Shopping Centre

3.3.2. Connecting page and html file

Input:

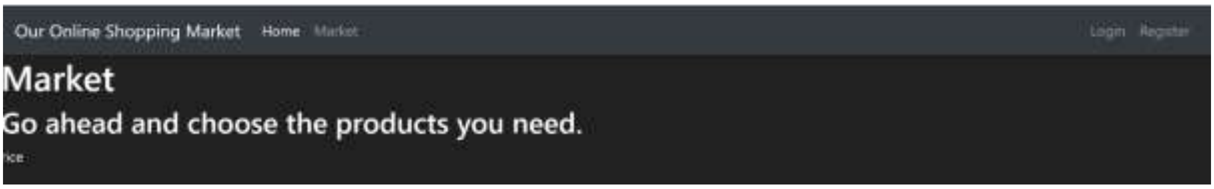
In the python file,

```
#add another page: market page
@app.route('/market')
def market():
    return render_template('market.html', item_name='rice')

#send some random data from Python to market.html: add key name i.e item_name
```

In the html file,

```
    <!--call the jinja syntax-->
    <p>{{ item_name }}</p>
```

Output/display: rice appears on the website created



Market

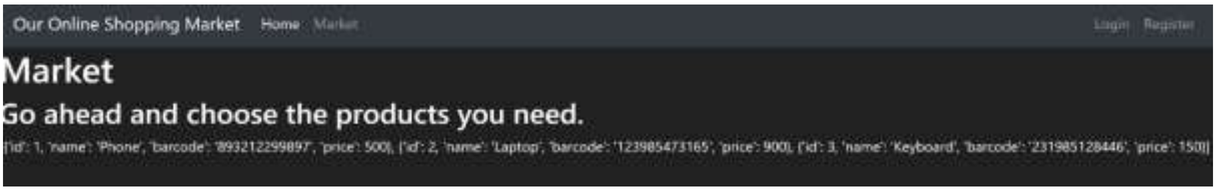
Go ahead and choose the products you need.

ice

Since lots of products are to be inserted, use list/dictionaries

```
#add list / dictionaries
def market():
    items = [
    {'id': 1, 'name': 'Phone', 'barcode': '893212299897', 'price': 500},
    {'id': 2, 'name': 'Laptop', 'barcode': '123985473165', 'price': 900},
    {'id': 3, 'name': 'Keyboard', 'barcode': '231985128446', 'price': 150}
    ]
#send some random data from Python to market.html: add key name ‘items’
    return render_template('market.html', items=items)
```

Make sure the key is the same in the market.html file:



Market

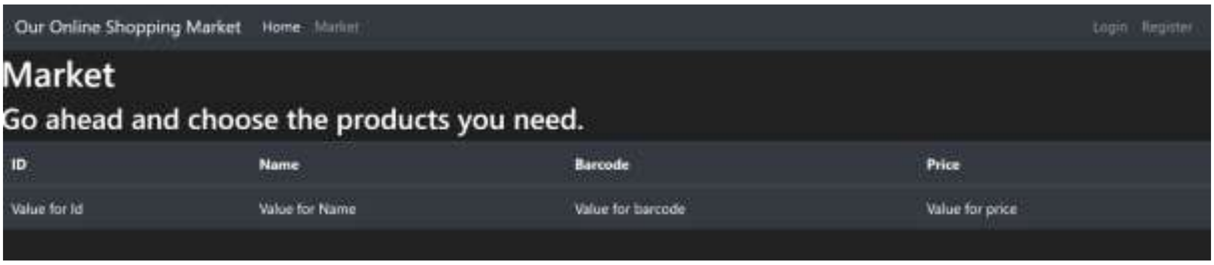
Go ahead and choose the products you need.

```
[{'id': 1, 'name': 'Phone', 'barcode': '893212299897', 'price': 500}, {'id': 2, 'name': 'Laptop', 'barcode': '123985473165', 'price': 900}, {'id': 3, 'name': 'Keyboard', 'barcode': '231985128446', 'price': 150}]
```

To have a better design, input

```
<!--class in an html file refers to bootstrap class, and it gives a nicer 
design-->
    <table class="table table-hover table-dark">
```

Output:



Market

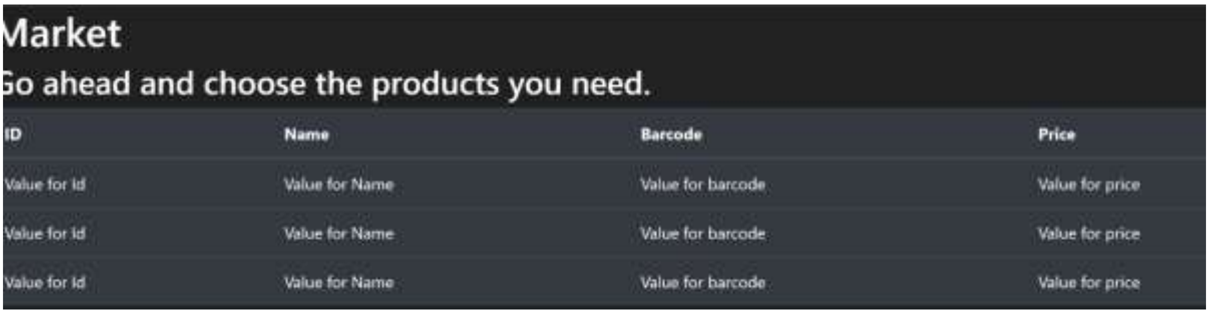
Go ahead and choose the products you need.

ID	Name	Barcode	Price
Value for Id	Value for Name	Value for barcode	Value for price

Input: Connecting html and python

```
<!--iterating the items/dictionaries, use the jinja template (allows us to 
access information that arrives from the route in a py file)-->
            <!--bring in a for loop in the html template-->
                {%for item in items%}
                <tr>
                    <td>Value for Id</td>
                    <td>Value for Name</td>
                    <td>Value for barcode</td>
                    <td>Value for price</td>
                </tr>
                {%endfor%}
```

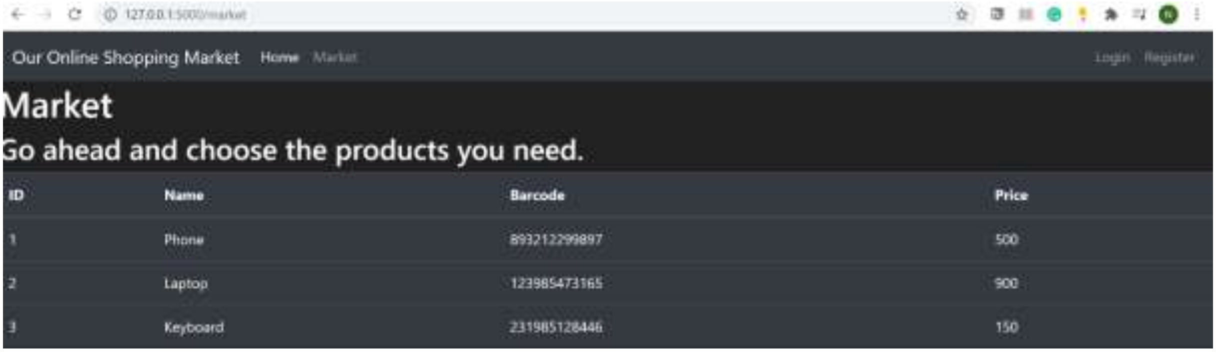
Output:


Market

Go ahead and choose the products you need.

ID	Name	Barcode	Price
Value for Id	Value for Name	Value for barcode	Value for price
Value for Id	Value for Name	Value for barcode	Value for price
Value for Id	Value for Name	Value for barcode	Value for price

The real value does not appear, therefore, replace <td>{{item.id}}</td>



Market

Go ahead and choose the products you need.

ID	Name	Barcode	Price
1	Phone	893212299897	500
2	Laptop	123985473165	900
3	Keyboard	231985128446	150

For the currency you wish to appear:

```
<td>{{item.price}}$</td>
```

3.3.3. Template Inheritance

```
<!--Template Inheritance: A base_template html file can be created to avoid 
repeating same codes.
     All new html files will inherit the attributes of the base_template.
    However, this is not done for this assignment-->
```

4. Test plan

Without a test plan, an app cannot be put on the market. There are also different types of testing of applications (Software Testing Fundamentals (STF), 18 September 2020; Homès, 2012). First, the remaining codes of this app must be written and tested as shown in the table below:

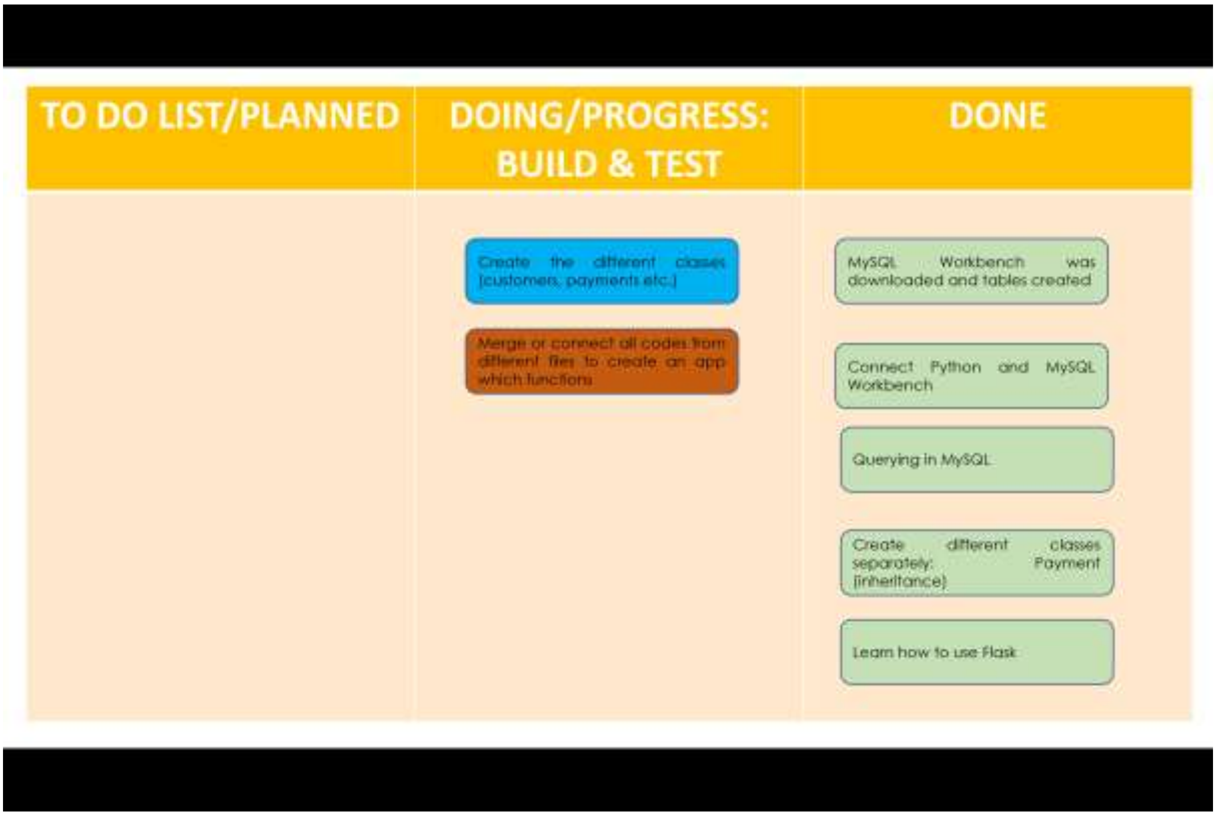
Start Date	Tasks or Features to write/test	Expected Results	Actual Results Pass-Fail Criteria	Comments	End Date
	A. UNIT/COMPONENT TESTING				
	Creating a database in MySQL	Creating a database in MySQL Workbench	Pass		
	Connecting database to Python	Connect the database to Python	Pass	Check files from submitted folder: products_dao.py	
	Querying	Add and Delete	Pass	products_dao.py	

		Product in the database from Python			
	Using Flask in Python	Importing Flask	Pass	Flask_server.py; market.py	
	Connecting Flask-python-html	Connecting the python file with the html file	Pass	market.py	
	Creating classes for the online app	Writing the codes for the different classes	Partially	-Shipping_detail.py Inheritance.py (for payment) -Having problems with creating codes to show composition (composition.py)	
B. INTEGRATION TESTING					
	App tested as a whole		Not done		
POSSIBLE TEST PLAN					
SYSTEM TESTING					
	Performance Test: Time it takes and memory used	NOT DONE – TO CHECK HOW TO DO THIS			
ACCEPTANCE TESTING					
	To test if the app is fit for use/put on the market	NOT DONE			

5. Review and Retrospective

From my point of view, this first attempt to create an e-commerce using object-oriented programming has been partially successful. However, the main challenge was in the planning and trying to find out what action must be done first. After reflection, it would be interesting to find out if it would have been better to create maybe one main.py file.

To sum up, connections among the files must be done and it must be improved and a better plan of work could be presented for this app or for future apps. The Kanban chart below is a tentative workplan to complete the work:



TO DO LIST/PLANNED	DOING/PROGRESS: BUILD & TEST	DONE
	Create the different classes (customers, payments etc.) Merge or connect all codes from different files to create an app which functions	MySQL Workbench was downloaded and tables created Connect Python and MySQL Workbench Querying in MySQL Create different classes separately: Payment (inheritance) Learn how to use Flask

6. Sources of inspiration/References

Apart from the lecture notes and Codio exercises available from Buckley, O. (2021), Object-oriented Information Systems OOIS_PCOM7E, University of Essex Online, delivered September and October 2021, the following websites and videos were consulted:

Brookshear, J. and Brylow, D. (2018) Computer Science - An Overview. 13th ed. Harlow: Pearson.

Bugti Jamal's videos. Available from : https://www.youtube.com/watch?v=o9TwipumGoU [Accessed 17 October 2021].

Codebasics (November 16, 2020) Grocery Store Application - 1. Overview – Python project tutorial. Available from: https://www.youtube.com/watch?v=0ZaC6JaNpic [Accessed 17 October 2021].

Codebasics (November 17, 2020) "Grocery Store Application - 2. Database Design – Python project tutorial". Available from: https://www.youtube.com/watch?v=VQsqdp_I9Ro [Accessed 17 October 2021].

Homès, B. (2012) *Fundamentals of Software Testing*. Great Britain: Wiley.

JimShapedCoding & freecodecamp.org (October 13, 2021) “Object Oriented Programming with Python - Full Course for Beginners”. Available from: https://www.youtube.com/watch?v=Ej_02ICOIgs [Accessed 24 October 2021]

JimShapedCoding (2021) Flask Full Series. Available from : http://www.jimshapedcoding.com/courses/Flask%20Full%20Series [Accessed 24 October 2021]

JP Infotech Projects (December 3, 2020) How to install Flask in Python – Windows 10 pip install flask. Available from: https://www.youtube.com/watch?v=wwG2lYcYltc [Accessed 22 October 2021]

Leach, R.J. (2016) *Introduction to Software Engineering*. 2nd ed. London: CRC Press/Taylor and Francis Group.

MrGoodChannel1. (July 22, 2019) *Python Debugging in Visual Studio Code part 1*. Available at: https://www.youtube.com/watch?v=RzfhZ0d3_Xw [Accessed 26 June 2021].

Oracle Corporation (2021) “MySQL Documentation”. Available from: https://dev.mysql.com/doc/ [Accessed 17 October 2021]

Paskhaver, B. (May 18, 2020) *Debugging Python in VSCode - 01 - Intro to Debugging in VSCode*. Available at: https://www.youtube.com/watch?v=KEdq7gC_RTA [Accessed 26 June 2021].

Programming with Mosh (March 20, 2019) “MySQL Tutorial for Beginners”. Available from: https://www.youtube.com/watch?v=7S_tz1z_5bA [Accessed 17 October 2021].

Programming with Mosh. (February 18, 2019) “Python Tutorial - Python for Beginners [Full Course]”. Available at: https://www.youtube.com/watch?v=_uQrJ0TkZlc [Accessed 30 June 2021].

Python, mysql and flask, https://roytuts.com/simple-shopping-cart-using-python-flask-mysql/

Software Testing Fundamentals (STF) “Test Plan” (September 18, 2020). Available at: https://softwaretestingfundamentals.com/test-plan/ [Accessed 26 June 2021].

University of Essex Online (2021). Codio exercises.

Yeoman, D. (2021) Five Types of Inheritance in Python. Available from: https://blog.finxter.com/understanding-inheritance-types-in-python/ [Accessed 17 October 2021]