Query_DB

May 14, 2021

[2]:
```
import mysql.connector

#I create the connection

db=mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="00000000",
    auth_plugin='mysql_native_password'

)

mycursor=db.cursor(buffered=True)

print(db)
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x7fa19d03bfd0>
```

[3]:
```
import time
import datetime
```

This function is made for insert new items in the table ITEMS, there are several input in wich I can insert all the information about a new article in the ecommerce site. If an article already exists, this will be an error for the database because already exists also the primary key and the constraitns about unique will be violated.

[ ]:
```
#this function will generate an item based on some inputs, and after it will␣
 ↪store into the DB

def insert_items():

    id_item=input('Enter id_item:')
    name_item=input("Enter name of item:")
    u_price=float(input('Enter price of unity:'))
    tot_quantity=input('Total quantity :')
    description=input('Enter little description:')
    id_supp=input('Enter id supplier:')
    print(id_supp)
```

```
    if id_supp:

        id_seller=None
    else:
        id_seller=input('Enter id seller:')
        id_supp=None

    id_place=input('Enter place where it is stored')

    if u_price>500:
        class_i='3-C'

    if u_price<=100 and u_price>0:
        class_i='2-C'

    if u_price>100 and u_price<500:
        class_i='1-C'

    if u_price<0:
        print('price negative!')

    sql='INSERT INTO ecommerce.
↪items(id_item,name_item,u_price,tot_quantity,description,id_class,id_supp,id_seller,id_plac
↪VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)'
  ␣
↪values=(id_item,name_item,u_price,tot_quantity,description,class_i,id_supp,id_seller,id_pla
    print(values)

    mycursor.execute(sql,values)

insert_items()
db.commit()
```

The function create_order() it will generate an order based on id_customer,id_item and quantity. I am assuming that there is a button in the GUI in wich this function is linked. So this funcrtion receive like parameters the information about the id_customer, id_item and quantity. After create a timestamp for the orders, the function will select the quantity aviable in the warehouse of the ecommerce site and compare with the quantity request. If the quantity exceed the quantity aviable, the function will give a message in wich say that the quantity in not avaible. Also this function will update the status of different tables releted to this. The function will update the ITEMS in order to decrease the quantity aviable, the table SALES in order to refresh with the new sells, and also check if there is a credit card inserted or if the credit card is new, in the last case the new credit card will be inserted.

[5]:
```
def create_order(id_customer,id_item,quantity,order_id,id_agent):

    #this is the main function, here I generate my order, and I create the␣
↪timestamp, also I passed some
    #parameters, to generate the order

    ts = time.time()
    date_ord = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d %H:%M:%S')


    id_item_str=str(id_item)
    quantity_str=str(quantity)
    order_id=str(order_id)
    id_customer=str(id_customer)
    id_agent=str(id_agent)

    mycursor.execute('SELECT u_price FROM ecommerce.items WHERE␣
↪id_item=%s'%id_item_str)

    myresult=mycursor.fetchall()

    price_unity=float(myresult[0][0])

    price_order=price_unity*quantity



    #here if the quantity exceed the quantity stored in the warehouse, the␣
↪function print that the quantity
    # exceed the stored quantity

    mycursor.execute('SELECT tot_quantity FROM ecommerce.items WHERE␣
↪id_item=%s'%id_item_str)

    myresult=mycursor.fetchall()

    quantity_tot=float(myresult[0][0])

    if quantity_tot < quantity:

        print('the item s quantity it is not aviable')

    else:

        update_quantity_item(quantity_tot,id_item_str,quantity)
```

```
        sql='INSERT INTO ecommerce.
↪orders(order_id,id_item,quantity,date_ord,price_order,id_customer,id_agent)␣
↪VALUES(%s,%s,%s,%s,%s,%s,%s)'

      ␣
↪values=(order_id,id_item_str,quantity,date_ord,price_order,id_customer,id_agent)

        mycursor.execute(sql,values)

        card_number=str(input('Enter card_number:'))

        insert_card(card_number,id_customer)


        create_sales(price_order,order_id,id_customer,card_number)

        db.commit()



#here i update the quantity stored in the warehouse

def update_quantity_item(quantity_tot,id_items,q):

    q_tot=quantity_tot-q

    q_tot=str(q_tot)

    sql='UPDATE ecommerce.items SET tot_quantity=%s WHERE id_item=%s'
    values=(q_tot,id_items)

    mycursor.execute(sql,values)



#check if exist alredy the card_number and the function will find the id of the␣
↪bank account(if alrady exist)

def insert_card(card_number,id_customer):


    condition=False
    mycursor.execute('SELECT bank_account_n FROM ecommerce.
↪customer_bank_account WHERE id_customer=%s'%id_customer)
    myresult=mycursor.fetchall()
```

```
    for i in myresult:

        for i in myresult:
            if card_number in i:
                condition=True




    if condition==True:
        #mycursor.execute('SELECT id_card_customer FROM ecommerce.
↪customer_bank_account WHERE bank_account_n=%s'%card_number )
        #myresult=mycursor.fetchall()
       # id_card_customer=myresult[0][0]
        return card_number


    else:
        sql_b_no='INSERT INTO ecommerce.
↪customer_bank_account(id_customer,bank_account_n)VALUES(%s,%s)'
        values=(id_customer,card_number)
        mycursor.execute(sql_b_no,values)




#Automatically will generate a sale

def create_sales(ammount,o_id,c_id,card_number):


    type_payment='Credit Card'

    ammount_1=ammount

    order_id=o_id

    mycursor.execute('SELECT id_card_customer FROM ecommerce.
↪customer_bank_account WHERE bank_account_n=%s'%card_number )
    myresult=mycursor.fetchall()
    id_card_customer=myresult[0][0]
```

```
    sql_payment='INSERT INTO ecommerce.payments(type_payment,id_card_customer)␣
↪VALUES(%s,%s)'
    values_payment=(type_payment,id_card_customer)

    mycursor.execute(sql_payment,values_payment)


    mycursor.execute('SELECT payment_id FROM ecommerce.payments ORDER BY␣
↪payment_id DESC LIMIT 1' )
    myresult=mycursor.fetchall()
    a=myresult[0][0]







    sql_sales='INSERT INTO ecommerce.sales(ammount,order_id,payment_id)␣
↪VALUES(%s,%s,%s)'
    values_sales=(ammount_1,order_id,a)

    mycursor.execute(sql_sales,values_sales)




#here there are the parameters passed to the main function

#create_order(5,131,1,9001,'130A')



db.commit()
```