

ASSIGNMENT-5

Frason Francis / 201903020 / 25

Develop GUI Application for E-commerce application use **(LO5)**

- 1) file, pickle, dictionary to show add, delete, update operations.
- 2) sqlite3 dictionary to show add, delete, update operations.

Theory : This is a E-Commerce Cryptocurrency Market Application where one can purchase and sell their Cryptocurrency. This programme also perform add, delete, update operation and uses Sqlite3 as database of the Application.

Code:

Main.py

```
1 from tkinter import *
2 from tkinter import messagebox, Menu
3 import requests
4 import json
5 import sqlite3
6
7 pycrypto = Tk()
8 pycrypto.title("CryptoMarket")
9 pycrypto.iconbitmap('favicon.ico')
10
11 con = sqlite3.connect('coin.db')
12 cursorObj = con.cursor()
13 cursorObj.execute("CREATE TABLE IF NOT EXISTS coin(id INTEGER PRIMARY KEY, symbol TEXT, amount INTEGER, price REAL)")
14 con.commit()
15
16 def reset():
17     for call in pycrypto.winfo_children():
18         call.destroy()
19
20     app_new()
21     app_header()
22     my_portfolio()
23
24 def app_new():
25     def clear_all():
26         cursorObj.execute("DELETE FROM coin")
27         con.commit()
28
29         messagebox.showinfo("Portfolio Notification", "Portfolio Cleared - Add New Coins")
30         reset()
31
32     def class_app():
```

```

11 def close_app():
12     pycrypto.destroy()
13
14     menu = Menu(pycrypto)
15     file_item = Menu(menu)
16     file_item.add_command(label="Clear Portfolio", command=clear_all)
17     file_item.add_command(label="Close App", command=close_app)
18     menu.add_cascade(label="File", menu=file_item)
19     pycrypto.config(menu=menu)
20
21 def get_my_portfolio():
22     api_request = requests.get("https://pro-api.coinmarketcap.com/v1/cryptocurrency/listings/latest?start=1&limit=300&convert=USD&CMC_PRO_API_KEY=387c3b38-387c3b38-387c3b38-387c3b38")
23     api = json.loads(api_request.content)
24
25     cursorObj.execute("SELECT * FROM coin")
26     coins = cursorObj.fetchall()
27
28     def Fort_value(amount):
29         if amount > 0:
30             return "green"
31         else:
32             return "red"
33
34     def insert_coin():
35         cursorObj.execute("INSERT INTO coin(symbol, price, amount) VALUES(?, ?, ?)", (symbol_txt.get(), price_txt.get(), amount_txt.get()))
36         con.commit()
37
38         messagebox.showinfo("Portfolio Notification", "Coin Added to Portfolio Successfully!")
39         reset()
40
41     def update_coin():
42         cursorObj.execute("UPDATE coin SET symbol=?, price=?, amount=? WHERE id=?", (symbol_update.get(), price_update.get(), amount_update.get(), portid_update.get()))

```

```

43     def update_coin():
44         cursorObj.execute("UPDATE coin SET symbol=?, price=?, amount=? WHERE id=?", (symbol_update.get(), price_update.get(), amount_update.get(), portid_update.get()))
45         con.commit()
46
47         messagebox.showinfo("Portfolio Notification", "Coin Updated Successfully!")
48         reset()
49
50     def delete_coin():
51         cursorObj.execute("DELETE FROM coin WHERE id=?", (portid_delete.get(),))
52         con.commit()
53
54         messagebox.showinfo("Portfolio Notification", "Coin Deleted From Portfolio")
55         reset()
56
57     total_pl = 0
58     coin_row = 1
59     total_current_value = 0
60     total_amount_paid = 0
61
62     for i in range(0, 300):
63         for coin in coins:
64             if api["data"][i]["symbol"] == coin[1]:
65                 total_paid = coin[2] * coin[3]
66                 current_value = coin[2] * api["data"][i]["quote"]["USD"]["price"]
67                 pl_percoin = api["data"][i]["quote"]["USD"]["price"] - coin[3]
68                 total_pl_coin = pl_percoin * coin[2]
69
70                 total_pl += total_pl_coin
71                 total_current_value += current_value
72                 total_amount_paid += total_paid
73
74     portfolio_id = Label(pycrypto, text=coin[0], bg="#F3F3F3", fg="black", font="Lato 12", borderwidth=2, relief="groove", padx="2", pady="2")

```

```

118
119 INSERT_COIN
120 symbol_tst = Entry(pycrypto, borderwidth=2, relief="groove")
121 symbol_tst.grid(row=coin_row+1, column=1)
122
123 price_tst = Entry(pycrypto, borderwidth=2, relief="groove")
124 price_tst.grid(row=coin_row+1, column=2)
125
126 amount_tst = Entry(pycrypto, borderwidth=2, relief="groove")
127 amount_tst.grid(row=coin_row+1, column=3)
128
129 add_coin = Button(pycrypto, text="Add Coin", bg="#142E54", fg="white", command=insert_coin, font="lato 12", borderwidth=2, relief="groove", padx=5, pady=5)
130 add_coin.grid(row=coin_row + 1, column=4, sticky=N+S+E+W)
131
132 UPDATE_COIN
133 portid_update = Entry(pycrypto, borderwidth=2, relief="groove")
134 portid_update.grid(row=coin_row+2, column=0)
135
136 symbol_update = Entry(pycrypto, borderwidth=2, relief="groove")
137 symbol_update.grid(row=coin_row+2, column=1)
138
139 price_update = Entry(pycrypto, borderwidth=2, relief="groove")
140 price_update.grid(row=coin_row+2, column=2)
141
142 amount_update = Entry(pycrypto, borderwidth=2, relief="groove")
143 amount_update.grid(row=coin_row+2, column=3)
144
145 update_coin_txt = Button(pycrypto, text="Update Coin", bg="#142E54", fg="white", command=update_coin, font="lato 12", borderwidth=2, relief="groove", padx=5, pady=5)
146 update_coin_txt.grid(row=coin_row + 2, column=4, sticky=N+S+E+W)
147
148 DELETE_COIN
149 portid_delete = Entry(pycrypto, borderwidth=2, relief="groove")

```

```

170 portfolio_id = Label(pycrypto, text="Portfolio ID", bg="#142E54", fg="white", font="lato 12 bold", padx=5, pady=5, borderwidth=2, relief="groove")
171 portfolio_id.grid(row=0, column=0, sticky=N+S+E+W)
172
173 name = Label(pycrypto, text="Coin Name", bg="#142E54", fg="white", font="lato 12 bold", padx=5, pady=5, borderwidth=2, relief="groove")
174 name.grid(row=0, column=1, sticky=N+S+E+W)
175
176 price = Label(pycrypto, text="Price", bg="#142E54", fg="white", font="lato 12 bold", padx=5, pady=5, borderwidth=2, relief="groove")
177 price.grid(row=0, column=2, sticky=N+S+E+W)
178
179 no_coins = Label(pycrypto, text="Coin Owned", bg="#142E54", fg="white", font="lato 12 bold", padx=5, pady=5, borderwidth=2, relief="groove")
180 no_coins.grid(row=0, column=3, sticky=N+S+E+W)
181
182 amount_paid = Label(pycrypto, text="Total Amount Paid", bg="#142E54", fg="white", font="lato 12 bold", padx=5, pady=5, borderwidth=2, relief="groove")
183 amount_paid.grid(row=0, column=4, sticky=N+S+E+W)
184
185 current_val = Label(pycrypto, text="Current Value", bg="#142E54", fg="white", font="lato 12 bold", padx=5, pady=5, borderwidth=2, relief="groove")
186 current_val.grid(row=0, column=5, sticky=N+S+E+W)
187
188 pl_coin = Label(pycrypto, text="P/L Per Coin", bg="#142E54", fg="white", font="lato 12 bold", padx=5, pady=5, borderwidth=2, relief="groove")
189 pl_coin.grid(row=0, column=6, sticky=N+S+E+W)
190
191 totalpl = Label(pycrypto, text="Total P/L With Coin", bg="#142E54", fg="white", font="lato 12 bold", padx=5, pady=5, borderwidth=2, relief="groove")
192 totalpl.grid(row=0, column=7, sticky=N+S+E+W)
193
194 app_exit()
195 app_header()
196 my_portfolio()
197 pycrypto.mainloop()
198
199 cursorObj.close()
200 con.close()

```

```

1 import requests
2 import json
3
4 api_request = requests.get("https://pro-api.coinmarketcap.com/v1/cryptocurrency/listings/latest?start=3&limit=5&convert=USD&apikey=837c1889c9b5")
5 api = json.loads(api_request.content)
6
7 print("-----")
8 print("-----")
9
10 coins = [
11     {
12         "symbol": "BTC",
13         "amount_owned": 2,
14         "price_per_coin": 3200
15     },
16     {
17         "symbol": "ETH",
18         "amount_owned": 100,
19         "price_per_coin": 2.05
20     }
21 ]
22
23 total_pl = 0
24
25 for i in range(0, 5):
26     for coin in coins:
27         if api["data"][i]["symbol"] == coin["symbol"]:
28             total_paid = coin["amount_owned"] * coin["price_per_coin"]
29             current_value = coin["amount_owned"] * api["data"][i]["quote"]["USD"]["price"]
30             pl_percoin = api["data"][i]["quote"]["USD"]["price"] - coin["price_per_coin"]
31             total_pl_coin = pl_percoin * coin["amount_owned"]

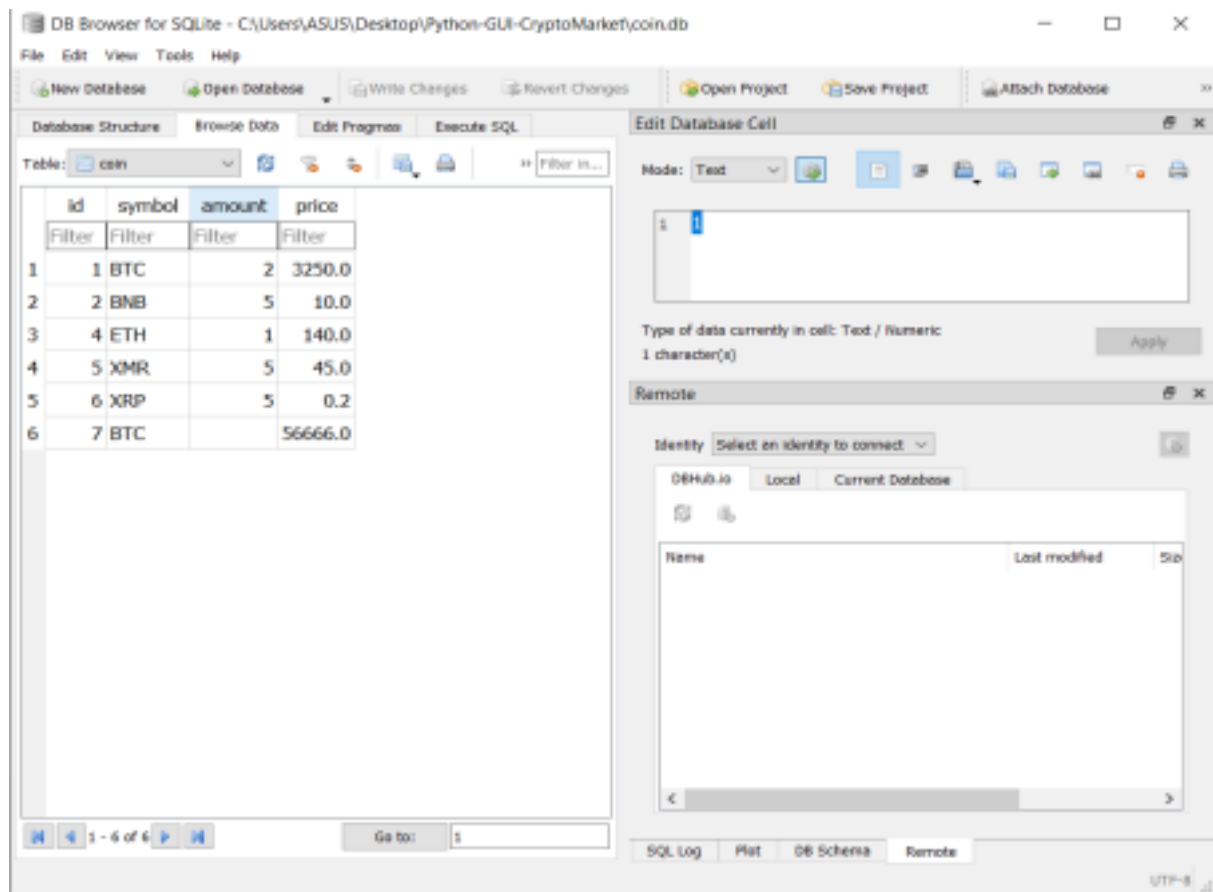
```

```

32     }
33 }
34
35 total_pl = 0
36
37 for i in range(0, 5):
38     for coin in coins:
39         if api["data"][i]["symbol"] == coin["symbol"]:
40             total_paid = coin["amount_owned"] * coin["price_per_coin"]
41             current_value = coin["amount_owned"] * api["data"][i]["quote"]["USD"]["price"]
42             pl_percoin = api["data"][i]["quote"]["USD"]["price"] - coin["price_per_coin"]
43             total_pl_coin = pl_percoin * coin["amount_owned"]
44
45             total_pl = total_pl + total_pl_coin
46
47             print(api["data"][i]["name"] + " - " + api["data"][i]["symbol"])
48             print("Price = ${0:.2f}".format(api["data"][i]["quote"]["USD"]["price"]))
49             print("Number Of Coins:", coin["amount_owned"])
50             print("Total Amount Paid:", "${0:.2f}".format(total_paid))
51             print("Current Value:", "${0:.2f}".format(current_value))
52             print("P/L Per Coin:", "${0:.2f}".format(pl_percoin))
53             print("Total P/L With Coin:", "${0:.2f}".format(total_pl_coin))
54             print("-----")
55
56 print("Total P/L For Portfolio:", "${0:.2f}".format(total_pl))

```

Database:



Output:

