TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Department of Computer Systems

Study Centre for Computer Systems

Toan Luong 182436MVEB

IAX0584 Programming III

Homework III

Homework III report

Supervisor René Pihlak

Declaration of originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. All works and major viewpoints of the other authors data from other sources of literature and elsewhere used for writing this paper have been referenced.

Name: Toan Luong

Date: 09/01/2020

Table of Contents

Task

Task variant: stockexchange_5A

Topic: Stock Exchange

Task description:

1. Foreign key constraints must be turned on.
2. Create tables DEALS and PRICES if not exist.
3. Make a menu that allows the user to execute commands until user signals that they want to exit
4. Allow only valid inputs. Warning if inputs are invalid.
5. Make commands for:
   a. Adding, modifying and deleting tables.
   b. Printing all action of a client based on user input. (Sort by date).
   c. Printing all action of a client for a given stock specified by user input. (Sort by date)
   d. Finding current portfolio for all clients. (Sort by client's last name and first name and stock's name).
   e. Finding current portfolio for a client. (Sort by stock's name)
   f. Finding profits for all clients.
   g. Finding profit for a client.
   h. Print out history of daily price changes.

Detailed description of the program

1. Functions:

char *mygetline(char *str, int maxlength);

Works similar to fgets(), however, does not require the stream. And also prevent newline problems. Do not need to flush the input buffer in case there is something, and also removed the newline character. Returns the address of the string.

int isLeap(int YYYY);

Take an integer year as argument and check if it is leap year or not.

int getDate(char **returnDate);

Requires a string pointer (char double pointer). The return value will be assigned to that pointer. Using ISO-8601 standard format <YYYY-MM-DD>. If every required input is correct, function will return SUCCEED code, if not a warning message will be display and INVALID_INPUT will be returned.

int doesExist(sqlite3 *conn,char* tableName, char* columnName,const void *value,
int dataType);

Requires a connection to DB as first parameter. The second and third parameter is the name of the table and the column name to look for. The fourth parameter is a pointer point to value what we want to search. Final parameter is according to original data type of fourth parameter.
dataType is 1 – if value is int*,

2 - if value is char*,

3 - if value is float*,

Any other will value of dataType will be warned and stop the program.

Example: SQL: “SELECT COUNT (*) FROM markets WHERE id_market = 1;”

My solution:

```
int i = 1;
doesExist(connection, “markets”,”id_market”,&i,1);
```

Checking existence of value in a table needed to be repeated a lot during the program, with doesExist() we can shorten the code.

void libmysqlite3 addingRecord(sqlite3 *conn);

The purpose of this function is showing a sub-menu for adding new record and redirect the flow to adding a record to clients, deals, prices, stocks and markets tables.

int libmysqlite3_addingRecord_clients(sqlite3 *conn);

int libmysqlite3_addingRecord_deals(sqlite3 *conn);

int libmysqlite3_addingRecord_prices(sqlite3 *conn);

int libmysqlite3_addingRecord_stocks(sqlite3 *conn);

int libmysqlite3_addingRecord_markets(sqlite3 *conn);

The function is created in order to adding a new record to one of the table. And checking the input of user. If input is invalid, a warning will be printed and return code will be INVALID_INPUT, else input will be insert in to table and return SUCCEED code.

NB!: if there is no market with id_market = 10, then you cannot add a new record to stocks table with id_market = 10. If you want to do so, make sure you create a market with id_market = 10 first. (checking Figure 1 for tables' relationship).

void libmysqlite3_deleteRecord(sqlite3 *conn);

A function showing sub-menu for deleting record and asking for traits of deleted record and redirecting.

void libmysqlite3_deleteRecord_clients(sqlite3 *conn,int id_client);

void libmysqlite3_deleteRecord_deals(sqlite3 *conn,int id_deal);

void libmysqlite3_deleteRecord_prices(sqlite3 *conn,int id_price);

void libmysqlite3_deleteRecord_stocks(sqlite3 *conn,int id_stock);

void libmysqlite3_deleteRecord_markets(sqlite3 *conn, int id_market);

The input which is provided in libmysqlite3_deleteRecord() will be used to mark the record which will be deleted in the above functions. Because SQLite will exec the command even if there is no such record without error, therefor no need to check the input from deleteRecord().

However, those delete table functions will call each other to make sure no record will violated foreign key constraint.

Eg: if you delete client with id 1 in clients table, all record with id_client = 1 will also be deleted in table deals. (checking Figure 1 for tables' relationship).

void libmysqlite3_updateRecord(sqlite3 *conn);

A function showing sub-menu for updating record and asking for traits of updated record and redirecting.

void libmysqlite3_updateRecord_clients(sqlite3 *conn,int id_client,
int forcedUpdate,int forcedUpdateTo);

void libmysqlite3_updateRecord_deals(sqlite3 *conn,int id_deal);

void libmysqlite3_updateRecord_prices(sqlite3 *conn,int id_price,
int forcedUpdate,int forcedUpdateTo);

void libmysqlite3_updateRecord_stocks(sqlite3 *conn,int id_stock,
int forcedUpdate,int forcedUpdateTo);

void libmysqlite3_updateRecord_markets(sqlite3 *conn, int id_market,

int forcedUpdate,int forcedUpdateTo);

Funtions use to update record according to it names, by the id value. However, there are two more parameters. forcedUpdate when turned off these functions work normally, when it is on functions will not allow user to choose column to update any more, the column is used to up date is defined by forcedUpdateTo,

Imagine when you change your ID document name, ownership of your property cannot be still under your old name (in such case you might lose your car, house, etc.). The ownership of these property will be changed to your new name.

Eg: market_name Tallinn changes id_market from 1 to 10, all of the stocks which are belong to id_market=1 in table stock will changes to 10. (checking Figure 1 for tables' relationship).

However, there are no need to used forcedUpdate and forcedUpdateTo parameters for libmysqlite3_updateRecord(); because its PK is not referenced to any table.

int libmysqlite3_find_client(sqlite3* conn);

int libmysqlite3_find_stock(sqlite3* conn);

These two functions are used to look for the id base on name or check if the id is valid or not. The return value would be the id or 0.

NB! There should be any record in any table with id is 0. Because the id is integer, therefor, there should be a value reserved to indicate NULL, in this case is 0.

int libmysqlite3_find_clientAction(sqlite3* conn,int withSpecifiedStock);

Function uses to control if user want to print all clients action or single client action depends on withSpecifiedStock flag , can be thought as a sub-menu for finding client action.

int libmysqlite3_print_clientAction(sqlite3 *conn,int id_client,int id_stock);

This function is used to print the action of client base on id_client. If the id_stock is non-zero, the result will be filtered by that id_stock.

int libmysqlite3_print_clientPortfolio(sqlite3 *conn,int id_client);

Printing all kind of stock specified by id_client. Invalid id_client will return INVALID_INPUT. Other case will be SUCCEED.

int libmysqlite3_print_allClientPortfolio(sqlite3 *conn)

function will search for all id_client present in clients table and call int libmysqlite3_print_clientPortfolio() for each id_client.

int libmysqlite3_find_singleClientPortfolio(sqlite3 *conn);

A sub-menu to redirect through libmysqlite3_find_client() and libmysqlite3_print_clientPortfolio () ;

int libmysqlite3_print_clientProfit(sqlite3 *conn,int id_client);

int libmysqlite3_find_singleClientProfit(sqlite3 *conn);

int libmysqlite3_find_allClientProfit(sqlite3 *conn);

These three function work with similar to the Portfolio family functions, but instead of printing Portfolio, they will print the Profit up to now.

int libmysqlite3_find_dailyChange_stock(sqlite3 *conn);

Function to finding and printing the change of stock with id_stock which will be specified inside it.

void libmysqlite3_print_change(float todayPrice,float yesterdayPrice);

Text displaying function for libmysqlite3_find_dailyChange_stock().

2. SQL querry:

“PRAGMA foreign_keys = ON;”

Turn foreign key constraint on.

```
"CREATE TABLE IF NOT EXISTS prices (
                    id_price INTEGER PRIMARY KEY,
                    id_stock INTEGER,
                    buy_price REAL,
                    sell_price REAL,
                    date text NOT NULL,"
                    FOREIGN KEY(id_stock) REFERENCES stocks(id_stock)
                    );
                    CREATE TABLE IF NOT EXISTS deals (
                    id_deal INTEGER PRIMARY KEY,
                    id_client INTEGER,
                    id_price INTEGER,
                    quantity INTEGER,
                    is_buy INTEGER NOT NULL"
                    FOREIGN KEY (id_client) REFERENCES clients (id_client),
                   FOREIGN KEY(id_price) REFERENCES prices(id_price));
```

Create tables prices and deals if they don’t exist.

```
INSERT INTO deals VALUES(?,?,?,?,?);
INSERT INTO clients VALUES(?,?,?);
INSERT INTO prices VALUES(?,?,?,?,?);
INSERT INTO stocks VALUES(?,?,?,?);
INSERT INTO markets VALUES(?,?,?);
```

Insert querries into different tables.

```
DELETE FROM clients WHERE id_client = ?;
DELETE FROM deals WHERE id_client = ?
```

DELETE FROM deals WHERE id_deal = ?;

DELETE FROM prices WHERE id_price = ?;

DELETE FROM deals WHERE id_price = ?;

DELETE FROM stocks WHERE id_stock = ?;

DELETE FROM markets WHERE id_market = ?;

Deleting record commands depends on different traits.

UPDATE clients SET id_client = ? WHERE id_client = ?;

UPDATE deals SET id_client = ? WHERE id_client = ?;

UPDATE clients SET first_name = ? WHERE id_client = ?;

UPDATE clients SET last_name = ? WHERE id_client = ?;

UPDATE deals SET id_deal = ? WHERE id_deal = ?;

UPDATE deals SET quantity = ? WHERE id_deal = ?;

UPDATE deals SET is_buy = ? WHERE id_deal = ?;

UPDATE prices SET id_price = ? WHERE id_price = ?;

UPDATE deals SET id_price = ? WHERE id_price = ?;

UPDATE prices SET buy_price = ? WHERE id_price = ?;

UPDATE prices SET sell_price = ? WHERE id_price = ?;

UPDATE prices SET date = ? WHERE id_price = ?;

UPDATE stocks SET id_stock = ? WHERE id_stock = ?;

UPDATE prices SET id_stock = ? WHERE id_stock = ?;

UPDATE stocks SET stock_name = ? WHERE stock_id = ?;

UPDATE stocks SET s_short_name = ? WHERE stock_id = ?;

UPDATE markets SET id_market = ? WHERE id_market = ?;

UPDATE stock SET id_market = ? WHERE id_market = ?;

UPDATE markets SET market_name = ? WHERE id_market = ?;

UPDATE markets SET m_short_name = ? WHERE id_market = ?;

Updating record’s column commands depends on different traits.

SELECT COUNT (*) FROM %s WHERE %s = %d;

SELECT COUNT(*) FROM stocks WHERE stock_name IS ? ;

SELECT COUNT(*) FROM clients WHERE first_name IS ? AND last_name IS ?;

SELECT COUNT (*) FROM clients ;

Counting record matching conditions command.

SELECT * FROM clients where id_client =?;

SELECT * FROM clients where id_client =?;

SELECT * FROM clients where id_client =?;

Selecting all columns from records that matched conditions.

SELECT id_price FROM prices WHERE id_stock = ?;

SELECT id_stock FROM stocks WHERE id_market = ?;

SELECT id_client FROM deals WHERE id_deal = ?;

SELECT id_price FROM deals WHERE id_deal = ?;

SELECT id_stock FROM prices WHERE id_price = ?;

SELECT id_market FROM stocks WHERE id_stock = ?;

SELECT id_stock FROM stocks WHERE stock_name IS ?;

SELECT id_client FROM clients WHERE first_name IS ? AND last_name IS ?;

SELECT id_stock, stock_name,s_short_name FROM stocks WHERE id_stock = ?

Selecting specified columns from records that matched conditions.

SELECT id_client FROM clients ORDER BY last_name ASC, first_name ASC;

SELECT date, sell_price,buy_price FROM prices WHERE id_stock = ? ORDER BY date ;

Selecting specified columns from records that matched conditions and sort by (date, last name,first name).

```
WITH
        LTBL AS ( SELECT * FROM deals LEFT JOIN clients USING (id_client))
       ,RTBL AS ( SELECT * FROM prices LEFT JOIN stocks USING (id_stock))
SELECT r.stock_name,r.id_stock,
         SUM ( CASE l.is_buy WHEN 1 THEN l.quantity
                                    ELSE -l.quantity END) AS share_holding
FROM LTBL l
LEFT JOIN RTBL r USING (id_price)
WHERE l.id_client = %d
GROUP BY r.id_stock
HAVING sum ( CASE l.is_buy WHEN 1 THEN l.quantity
                                       ELSE -l.quantity END) > 0
ORDER by r.stock_name ASC;
```

Joining deals & clients (LeftTable) and prices & stocks (RightTable).

Joining LeftTable & RightTable (ResultTable).

Selecting stock_ name, id_stock columns from ResultTable, create new column name share_holding which is a sum of all quantity (add for buy and subtract for sell) for each type of id_stock (GROUP BY r.id_stock), for a specified client (WHERE l.id_client = %d),

And showing those selecting (as new now) only if the sum is larger than 0.

Final table is sorted by stock_name ascending.

Finding portfolio for a specified client.

```
WITH tempTBL
                AS (  SELECT p.date,d.id_client,d.quantity,p.id_stock,
                        CASE d.is_buy WHEN 1 THEN -p.buy_price*d.quantity
                                ELSE p.sell_price*d.quantity END
                                AS spend_earn_by_stock_by_date
                    FROM deals d
                    LEFT JOIN prices p USING (id_price)
                    WHERE d.id_client = %d
                    ORDER BY p.date ASC )
SELECT id_client,sum(spend_earn_by_stock_by_date) as total_spend_earn
FROM tempTBL;
```

Making new table (tempTBL) has date,id_client (to double check),quantity,id_stock, and spend_earn_by_stock_by_date(which is money earn or spend by each stock by a single day *) for a specified client. Then sum the all money that earn or spend by that client.

Finding profit for a client.

* when column is_buy = 1, spend_earn_by_stock_by_date = -buy_price * quantity

is_buy = 0, spend_earn_by_stock_by_date = sell_price * quantity

the negative value indicate spending money.

Eg: 2019-12-31 id_stock= 100 has sell_price = 3 and buy_price = 4

id_client=101 buys 1000 id_stock=100 on 2019-12-31

then id_client=101 sells 50 id_stock1 on 2019-12-31

The tempTBL will show :

date	id_client	quantity	id_stock	spend_earn
2019-12-31	101	1000	100	-1000*4
2019-12-31	101	50	22	50*3

And the resultTable will be:

id_client	total
101	-1000*4 +50 *3 = -3850

Which mean client 101 lost 3850$ in total.

3. Database schema

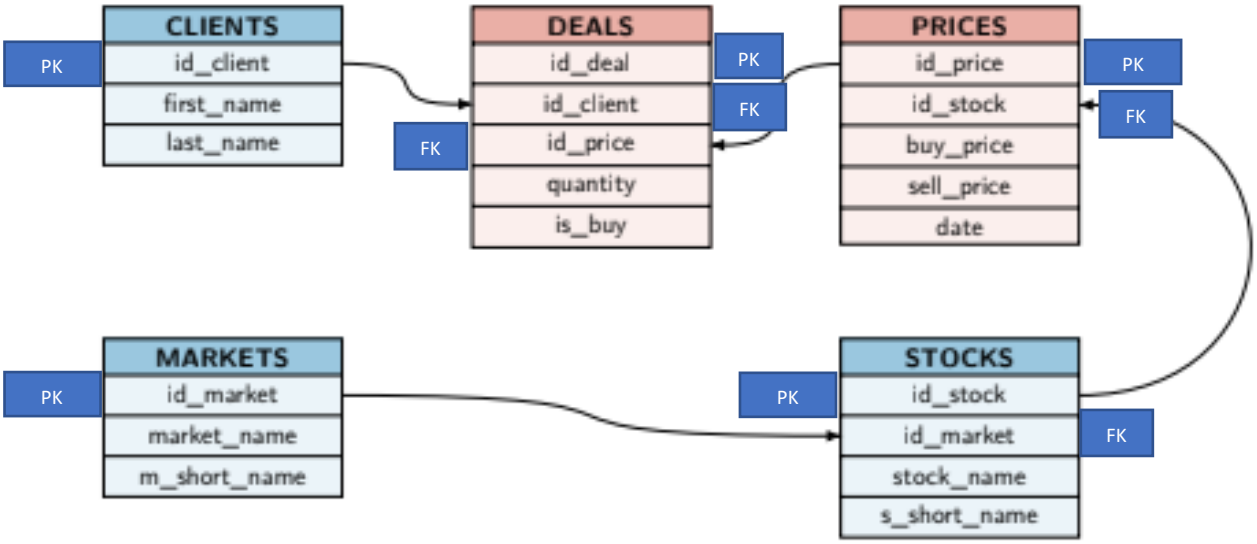

Figure 1

Screenshot

```
Option:
0-Quit
1-Adding Record
2-Delete Record
3-Update
4-Find client action.
5-Find client action with for a specified stock.
6-Find all client portfolio.
7-Find client portfolio.
8-Find all clients profit.
9-Find single client profit.
10-Find history of daily price change for a stock
11-Clear terminal
>
```

Main menu

```
Option:
0-Quit
1-Adding Record
2-Delete Record
3-Update
4-Find client action.
5-Find client action with for a specified stock.
6-Find all client portfolio.
7-Find client portfolio.
8-Find all clients profit.
9-Find single client profit.
10-Find history of daily price change for a stock
11-Clear terminal
>10
        (1- Using id_stock) (2- Using stock_name)
        >1
        Enter id_stock:
        >1
History of price change of: Arco Vara ,ABRV:ARC1T (id_stock:1)
   Date   |              Sell                |        Buy
2012-12-01|         Starting:  2.10€         | Starting: 1.20€
2012-12-05| 2.20€ increases by  0.10€(  4.76%)| 2.50€ increases by  1.30€(108.33%)
2012-12-10| 4.00€ increases by  1.80€( 81.82%)| 3.10€ increases by  0.60€( 24.00%)
2012-12-10|12.90€ increases by  8.90€(222.50%)|12.50€ increases by  9.40€(303.23%)
2012-12-15|11.20€ decreases by  1.70€( 13.18%)|11.30€ decreases by  1.20€(  9.60%)
2012-12-21|10.20€ decreases by  1.00€(  8.93%)|11.40€ increases by  0.10€(  0.88%)
2012-12-22|11.20€ increases by  1.00€(  9.80%)|11.50€ increases by  0.10€(  0.88%)
2012-12-23| 7.57€ decreases by  3.63€( 32.41%)|11.80€ increases by  0.30€(  2.61%)
2012-12-24| 3.24€ decreases by  4.33€( 57.20%)|10.30€ decreases by  1.50€( 12.71%)
2012-12-25|12.58€ increases by  9.34€(288.27%)|11.80€ increases by  1.50€( 14.56%)
2012-12-26| 2.53€ decreases by 10.05€( 79.90%)| 8.70€ decreases by  3.10€( 26.27%)
Option:
0-Quit
1-Adding Record
```

Finding history of stock by id_stock

```
Option:
0-Quit
1-Adding Record
2-Delete Record
3-Update
4-Find client action.
5-Find client action with for a specified stock.
6-Find all client portfolio.
7-Find client portfolio.
8-Find all clients profit.
9-Find single client profit.
10-Find history of daily price change for a stock
11-Clear terminal
>10
        (1- Using id_stock) (2- Using stock_name)
        >2

Enter stock_name:
        >Arco Vara
History of price change of: Arco Vara ,ABRV:ARC1T (id_stock:1)
   Date    |                Sell                 |         Buy
2012-12-01|          Starting:  2.10€           | Starting: 1.20€
2012-12-05| 2.20€ increases by  0.10€(  4.76%)| 2.50€ increases by  1.30€(108.33%)
2012-12-10| 4.00€ increases by  1.80€( 81.82%)| 3.10€ increases by  0.60€( 24.00%)
2012-12-10|12.90€ increases by  8.90€(222.50%)|12.50€ increases by  9.40€(303.23%)
2012-12-15|11.20€ decreases by  1.70€( 13.18%)|11.30€ decreases by  1.20€(  9.60%)
2012-12-21|10.20€ decreases by  1.00€(  8.93%)|11.40€ increases by  0.10€(  0.88%)
2012-12-22|11.20€ increases by  1.00€(  9.80%)|11.50€ increases by  0.10€(  0.88%)
2012-12-23| 7.57€ decreases by  3.63€( 32.41%)|11.80€ increases by  0.30€(  2.61%)
2012-12-24| 3.24€ decreases by  4.33€( 57.20%)|10.30€ decreases by  1.50€( 12.71%)
2012-12-25|12.58€ increases by  9.34€(288.27%)|11.80€ increases by  1.50€( 14.56%)
2012-12-26| 2.53€ decreases by 10.05€( 79.90%)| 8.70€ decreases by  3.10€( 26.27%)
Option:
```

Finding history of stock by stock_name

```
7-Find client portfolio.
8-Find all clients profit.
9-Find single client profit.
10-Find history of daily price change for a stock
11-Clear terminal
>8
Keit Anvelt (ClientID: 15) does not have any profit!
Mailis Helme (ClientID: 10) does not have any profit!
Hanno Herkel (ClientID: 18) does not have any profit!
Valdo Herkel (ClientID: 47) does not have any profit!
Mihhail Kruuse (ClientID: 27) does not have any profit!
Anne Lauri (ClientID: 44) does not have any profit!
Artur Lauri (ClientID: 46) does not have any profit!
Marika Lauri (ClientID: 33) does not have any profit!
Urmas Lauri (ClientID: 22) does not have any profit!
Mailis Lenk (ClientID: 48) does not have any profit!
Mart Ligi (ClientID: 34) does not have any profit!
Toan Luong (ClientID: 100) earns 648206.27 €!
Eiki Lutsepp (ClientID: 37) does not have any profit!
Mihhail Lutsepp (ClientID: 40) does not have any profit!
Heimar Michal (ClientID: 36) does not have any profit!
Kristen Michal (ClientID: 3) earns 166.00 €!
Mailis Michal (ClientID: 32) does not have any profit!
Marika Michal (ClientID: 23) does not have any profit!
Taavi Mikser (ClientID: 49) does not have any profit!
Juhan Nestor (ClientID: 24) does not have any profit!
Marika Ossinovski (ClientID: 9) does not have any profit!
Mihkel Padar (ClientID: 39) does not have any profit!
Rein Padar (ClientID: 4) does not have any profit!
Kristen Paet (ClientID: 43) does not have any profit!
Sven Paet (ClientID: 28) does not have any profit!
Urve Paet (ClientID: 16) does not have any profit!
Ivari Parts (ClientID: 7) does not have any profit!
Andres Pentus-Rosimannus (ClientID: 11) does not have any profit!
Jevgeni Pentus-Rosimannus (ClientID: 45) does not have any profit!
Mailis Purga (ClientID: 30) does not have any profit!
Mihhail Purga (ClientID: 25) does not have any profit!
Maris Randpere (ClientID: 5) does not have any profit!
Jevgeni Ratas (ClientID: 20) does not have any profit!
Ants Reps (ClientID: 19) does not have any profit!
Juhan Savisaar (ClientID: 35) does not have any profit!
Mihkel Savisaar (ClientID: 17) does not have any profit!
Kadri Seeder (ClientID: 12) does not have any profit!
Rein Seeder (ClientID: 6) does not have any profit!
Urmas Seeder (ClientID: 26) does not have any profit!
```

Finding profit of all client

```
Option:
0-Quit
1-Adding Record
2-Delete Record
3-Update
4-Find client action.
5-Find client action with for a specified stock.
6-Find all client portfolio.
7-Find client portfolio.
8-Find all clients profit.
9-Find single client profit.
10-Find history of daily price change for a stock
11-Clear terminal
>9
        (1- Using id_client) (2- Using client first and last name)
        >2
        Enter first_name:
        >Toan
        Enter last_name:
        >Luong
Toan Luong (ClientID: 100) earns 648206.27 €!
Option:
0-Quit
```

Finding profit of a specified client

```
6-Find all client portfolio.
7-Find client portfolio.
8-Find all clients profit.
9-Find single client profit.
10-Find history of daily price change for a stock
11-Clear terminal
>6
ClientID|First Name|Last Name
  15    |Keit      |Anvelt
================================
Client is not holding any stock share at the moment!
ClientID|First Name|Last Name
  10    |Mailis    |Helme
================================
Client is not holding any stock share at the moment!
ClientID|First Name|Last Name
  18    |Hanno     |Herkel
================================
Client is not holding any stock share at the moment!
ClientID|First Name|Last Name
  47    |Valdo     |Herkel
```

Finding porfolio of all clients

References

https://www.sqlite.org/index.html

https://stackoverflow.com/

https://www.youtube.com/channel/UCCTVrRB5KpIiK6V2GGVsR1Q kudvenkat(PragimTechnologies)