CS480 – Course project

Summer 2020

Database: **Personal Finance Management System**

Description:

The overall idea of this project is to create a database to store personal financial information of an individual(s).

- An individual **user** is characterized by a unique ***user id*** and ***name, address, phone number***(s), (and potentially other personal information). An individual's **income** and **expense** are identified by (a unique ***transaction ID, description, date, amount, tags***). Tags provide the user with the ability to look back at past incomes and expenses, study the area where they were incurred, and analyze important/wasteful expenses, leading to further financial insight.

- The **bank balance** of the individual (on a particular day) is characterized by the unique pair (***ID, date***) and contains the ***balance*** amount. The bank balance may be updated multiple times on a day hence we need two attributes to characterize a unique primary key.

- A particular **stock** is characterized by its (unique ***symbol*** (e.g. AAPL - Apple Inc.) , ***current price***). A user is linked to each stock that they buy/sell through a **transaction** entity where each tuple is transaction characterized by a unique ***transaction id*** and described otherwise by a ***user id, symbol, quantity, buying price***, and ***total value*** of the transaction.

- Each tuple in the individual's **portfolio** is characterized by a unique (***user id,symbol***) pair and the stock's (***buy price , quantity , total investment, current price, equity***). The portfolio records how much money the individual has invested in each stock.

- Each **asset** and **liability** is characterized by a unique (***user id, asset name/liability name***) pair, besides the ***amount*** value of that asset/liability. A date is not needed here since the assets and liabilities are reported at the time that they are accessed, and not on a specific date.

- Finally, the **balance sheet** is an ongoing record of the overall balance of the individual which is a sum of the individual's bank balance, cash on hand, all assets, and all liabilities. Each record in the balance sheet is characterized by a unique (***user ID, date***) pair and also contains the ***balance***. The total balance too can change multiple times within a day and thus must be identified by a unique ID, rather than a date.

Project Part 2 – CRUD (Create, read, update, and delete)

Deadline: July 18, 2020

List of entities:

1. User
2. Income
3. Expense
4. Liability
5. Balance Sheet
6. Portfolio

Based on the Demo (Part 1), implement the following functionality using Java and SQL with necessary GUI interfaces.

1. Insert/delete/update/read a **user** (all attributes except the user id). The user id should be generated by the system automatically using MySQL autoincrement.

2. Insert/delete/update/read an **income** (all attributes except the income id). The income id should be generated by the system automatically using MySQL autoincrement.

3. Insert/delete/update/read an **expense** (all attributes except the expense id). The expense id should be generated by the system automatically using MySQL autoincrement.

4. Insert/delete/update/read an **account** (all attributes except the account id). The account id should be generated by the system automatically using MySQL autoincrement.

5. Insert/delete/update/read an **asset** (all attributes). The asset name must be chosen from a dropdown list.

6. Insert/delete/update/read an **liability** (all attributes). The liability name must be chosen from a dropdown list.
7. Insert/delete/update/read a **balance sheet** (all attributes).
8. Insert/delete/update/read a **stock** (all attributes).
9. Insert/delete/update/read a **transaction** (all attributes except the transaction id and transaction date and time). The transaction id, date, and time should be generated by the system automatically using MySQL autoincrement.
10. Insert/delete/update/read a **portfolio** (all attributes).

Project Part 3 – Queries

Deadline: August 1, 2020

Based on the Demo, implement the following functionality using Java and SQL with necessary GUI interfaces.

Trivial Queries:

1. List all users members.
2. List all stocks owned by a specific user.
3. List all liabilities.

Non-trivial Queries:

1. List all stocks currently yielding a profit > x%
2. List a user's expenses which have a certain tag (e.g. 'carryout'), and are incurred in a certain month.
3. List the difference between total incomes and total expenses (by a user), as a percentage, for each month in a year.
4. List all months in the year where a user's balance (from balance sheet) has decreased from the previous month.
5. List all users whose 'credit card' liability has been increasing for the past six months.