In this project, you will design and implement an Investment Portfolio database.

The conceptual data model for investment portfolio data features users, accounts, trades, instruments and positions. A user has a unique username and may have other attributes like name. An account has a unique number, cash balance, investment value and total value. A trade is uniquely identified by an id and can be either a buy transaction or a sell transaction. Other trade attribute types include a trade date time, number of shares, price per share and total amount. An instrument has a unique symbol and a current quote. Stocks, mutual funds and exchange-traded funds (ETFs) are all types of instruments supported in this example. While a user can open many accounts, each account must belong to exactly one user. Similarly, an account can place many trades and an instrument can participate in many trades, but a trade is always associated with only one account and one instrument. Finally, an account may have many positions and an instrument can be owned by many accounts. Each position in a particular account is described by an instrument symbol, quantity and current value.

Note that there are four derived attribute types, namely investment value, total value, current value and amount. Derived attribute values are computed based on other attribute values. For example, a current position value is computed by multiplying a quantity by a quote for a particular instrument. An account investment value is the sum of all current position values. And an account total value is the sum of a cash balance and an investment value. Last but not least, a trade amount is the product of a price and a number of shares. In general, while some derived attribute values can be stored in a database, others can be dynamically computed by an application.

First, you need to design a conceptual model of the database and draw an **ER diagram** that can capture the information needed for this database. In the ER diagram, you need to model the data stored in the database as entity sets and relationship sets with cardinality and participation constraints, show attributes, and identify primary key for each entity set. You should use the same notation in the textbook.

Next, you should translate your ER diagram into a set of relational tables with primary key and foreign key constraints (if any) indicated by writing **SQL create table statements**. You should populate the database using **insert into statements** with at least five tuples per table (except instruments table) with fabricated data or data downloaded from the Internet (Please ensure that each query below has **at least one** result).

Then, perform the following **queries** (each query should be answered by a **single** SQL statement):

**Queries**

1. List all usernames of users in ascending order.
2. Find the name of the user whose account number is 122000247.
3. List the tradeIDs that were generated after 2023-03-10 09:00:00 and their corresponding usernames. List your result in the ascending order of the trade date time.
4. Find the instruments that have been owned by at least two users.
5. Find the user who holds the maximum account total value.
6. List the username and instruments of the user who has bought maximum shares of instrument in 2022.
7. Find the user who has bought every instrument.
8. Insert a new user.
9. Increase $1 to those instruments whose current quote is lower than $25.
10. Delete users without trade records.

**Materials to Turn In**

1.Put the ER diagram in a PDF file. You can either draw by hand or use Microsoft Word, etc.

2.Put all create table and insert into statements in one “your-name-table.sql” file and all 10 queries in one "your-name-query.sql" file. All 10 queries should be implemented in the given order.

3.Submit the PDF file and the two .sql files by 11:59PM, December 21, 2023 from Blackboard.

**Things to remember**

There will be 5 points for the ER diagram, 5 points for the create table and insert into statements, and 1 point for each query.

The two .sql files must run correctly on cs-oracle.cs.uml.edu to receive credits.

Create table statements must include primary key and foreign key constraints (if any). Queries will be graded on correctness. Few points may be deducted for bad style like use of complicated queries which are difficult to understand.

Late submissions will not be accepted.

You should not discuss the project with classmates not in your team.

**The project is an individual project. Plagiarism is prohibited. If found, all the students involved will get zero credit for the project.**