PSE: Poughkeepsie Stock Exchange

Database Design Plans

Database Designer: Michael Figueiredo

Fall 2014

Table of Contents:

Executive Summary:

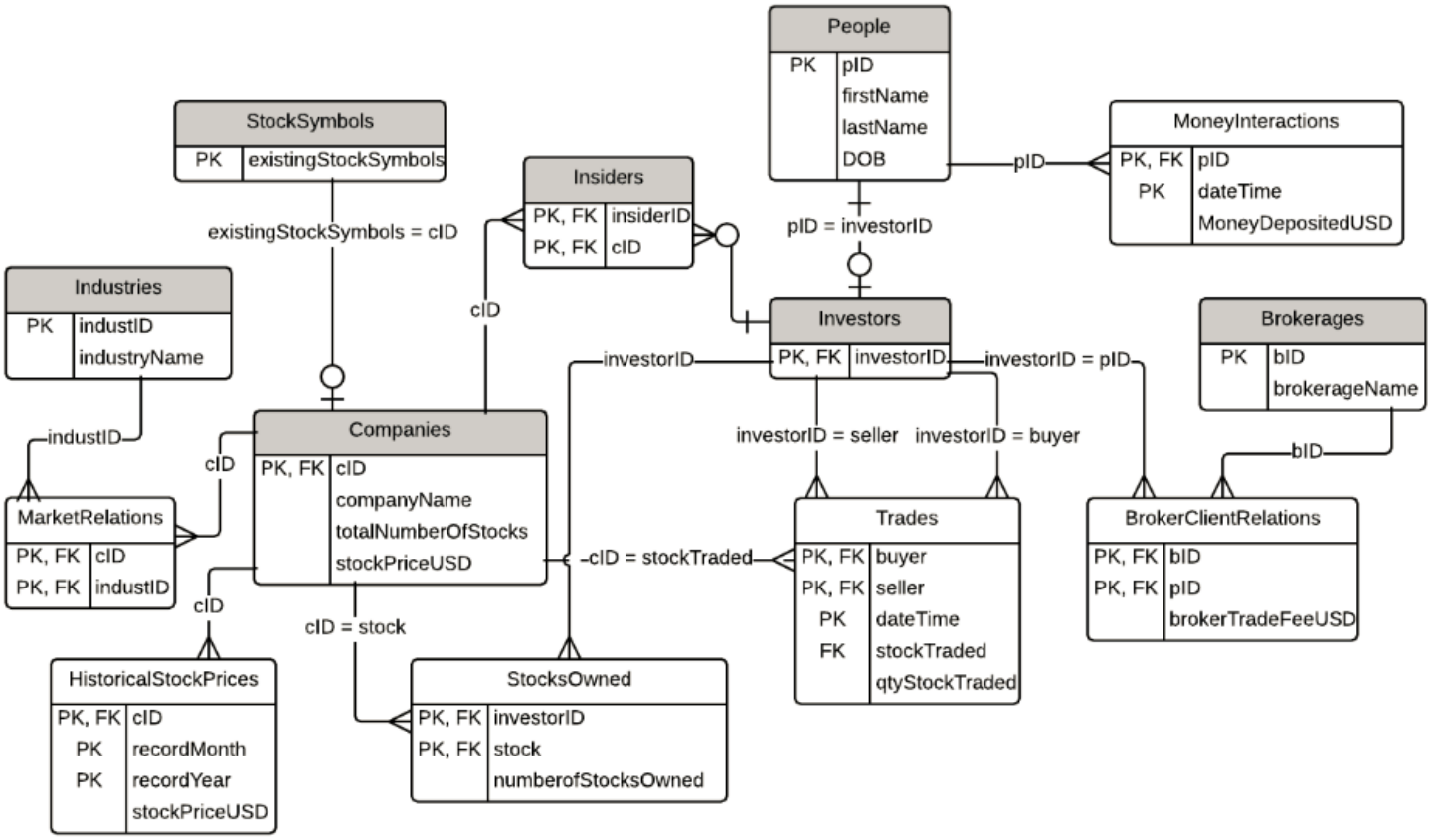
Overview:

In a strange and foreign future, much different than the world we live in today, it is very possible that stock exchanges will emerge in every major city in the United States (or they won't… we're really not sure… Ok they probably won't). In any case, why not be prepared to handle such an absurd prospect by planning ahead and implementing our very own database that can support the needs and queries of a future application system that can manage stock market transactions of people seeking to invest on this new stock exchange? Poughkeepsie, NY, the quaint and wonderful jewel of the Hudson River Valley would of course be high up on the list of major cities to require such a stock exchange, so it is essential that we prepare now so as to maximize the potential gain of such a shining opportunity that our future does (or most likely does not) hold.

Objectives:

- Allow for buying and selling of stocks by investors for stocks available on the PSE
- Allow for stock brokers to have clients and manage stocks on the exchange for a fee per trade
- Allow companies to exist within various industries, thus segmenting companies into their subsequent businesses and allow investors to buy stock in these various lines of business independently
- Have fun with it
- Don't spend too much money implementing the necessary database backend for the stock exchange system
    - Have a random college student design the database for free

Entity-Relationship Diagram:



Tables:

People:

Purpose:

This table is designed to uniquely represent persons in a generic sense, storing their names and dates of birth as well as a unique identifier for each person. It exists for the purpose of extending the definition of a person to investors and company insiders in subsequent tables.

SQL Create Statement:

```
CREATE TABLE People (
  pID                   char(10) not null,
  firstName             text not null,
  lastName              text not null,
  DOB                   date not null,
 primary key(pID)
);
```

Functional Dependencies:

pID → firstName, lastName, DOB

Sample Data:

	pid character(10)	firstname text	lastname text	dob date
1	p000000000	Mike	Figs	1901-01-01
2	p000000001	Teddy	Roosevelt	1858-10-27
3	p000000002	Alan	Prime	1000-12-31
4	p000000003	Baby	NewYear	1940-01-01
5	p000000004	Johnny	Moneybags	1977-04-15
6	p000000005	Wendy	Ecofriendly	1985-06-20
7	p000000006	Sandra	Steel	1895-05-12
8	p000000007	Zeke	Techsavvy	1980-08-05
9	p000000008	Harry	Hungryman	1979-03-17
10	p000000009	Thomas	Thrifty	1954-07-25

Investors:

Purpose:

This table stores all people who have invested in the Poughkeepsie Stock Exchange. The table stores only the unique ID of all investors as queries can be used to calculate information that is more variable in nature such as total stock values or liquid assets in the stock exchange (see 'Calculating Sum of Client's Stock Values' query in the Reports section of this document).

SQL Create Statement:

```
CREATE TABLE Investors (
  investorID          char(10) not null references People(pID),
 primary key(investorID)
);
```

Functional Dependencies:

investorID →

Sample Data:

	investorid character(10)
1	p000000000
2	p000000001
3	p000000002
4	p000000003
5	p000000004
6	p000000005
7	p000000006
8	p000000007
9	p000000008

StockSymbols:

Purpose:

This table stores all existing stock/ticker symbols on the Poughkeepsie Stock Exchange so as to be referenced in the Companies table as the unique primary key (cID) of each company that can be invested in on this stock exchange.

SQL Create Statement:

```
CREATE TABLE StockSymbols (
  existingStockSymbols text not null,
 primary key(existingStockSymbols)
);
```

Functional Dependencies:

existingStockSymbols →

Sample Data:

	existingstocksymbols text
1	RCPR
2	APDR
3	CDR
4	RR
5	CB
6	MSFA
7	TBE
8	RYGC

Companies:

Purpose:

This table stores all companies that can be invested in on the Poughkeepsie Stock Exchange, identified by their stock/ticker symbol so as to allow for the same company to have multiple stocks on the market for different aspects of their business. The table also stores the total number of each stock that exists in the market and the current value of this stock in US dollars.

SQL Create Statement:

```
CREATE TABLE Companies (
  cID                 text not null references StockSymbols(existingStockSymbols),
  companyName         text not null,
  totalNumberOfStocks integer default 0,
  stockPriceUSD       numeric(8,2) default 0.00,
 primary key(cID)
);
```

Functional Dependencies:

cID → companyName, totalNumberOfStocks, stockPriceUSD

Sample Data:

	cid text	companyname text	totalnumberofstocks integer	stockpriceusd numeric(8,2)
1	APDR	AlwaysPrepared Diner Restaurant	10000	12.74
2	CDR	Castle Diner Restaurant	10000	13.99
3	CB	College Brewery	20000	20.45
4	MSFA	MovieStreamingForAll	15000	16.25
5	RCPR	Random College Pizza Restaurant	4000	9.26
6	RYGC	Recycle-Your-Goods Center	8000	11.11
7	RR	Riverside Railroad	400	2.13
8	TBE	Textbook Emporium	2000	3.11

Insiders:

Purpose:

This table is meant to store all investors who are considered insiders in a specific company, a term that varies based upon the company in question, but normally refers to an investor or stockholder that holds a significant percentage of the available stock on the market and/or has an executive role in the company. Though this table only stores the investor's ID and the stock symbol of the company in which the investor is an insider, calculations from other tables such as the StocksOwned table and Companies table will allow users to retrieve other pieces of information such as the percentage of the stock owned or the value of each insider's investments in each related company.

SQL Create Statement:

```
CREATE TABLE Insiders (
  insiderID          char(10) not null references Investors(investorID),
  cID                text not null references Companies(cID),
 primary key (insiderID, cID)
);
```

Functional Dependencies:

insiderID, cID →

Sample Data:

	insiderid character(10)	cid text
1	p000000000	CDR
2	p000000000	TBE
3	p000000001	RR
4	p000000002	TBE
5	p000000004	CB
6	p000000005	RYGC

Trades:

Purpose:

This table stores the history of stock transactions between investors, including the IDs of the buyer and seller as well as the date and time of the transaction, the stock being traded, and the amount of the stock being traded. Having a history of transactions is important for keeping track of the movements of stocks and for legal documentation purposes, but in future implementations of a much more realistic stock exchange system, these trades could be tracked and used to calculate changes in stock prices as well.

SQL Create Statement:

```
CREATE TABLE Trades (
  buyer               char(10) not null references Investors(investorID),
  seller              char(10) not null references Investors(investorID),
  dateTime            timestamp not null,
  stockTraded         text not null references Companies(cID),
  qtyStocksTraded     integer not null,
 primary key(buyer, seller, dateTime)
);
```

Functional Dependencies:

buyer, seller, dateTime → stockTraded, qtyStockTraded

Sample Data:

	buyer character(10)	seller character(10)	datetime timestamp without time zone	stocktraded text	qtystockstraded integer
1	p000000002	p000000008	2014-07-12 11:11:11	CDR	125
2	p000000006	p000000001	2014-07-12 14:06:30	RR	16
3	p000000004	p000000007	2014-07-12 21:12:00	MSFA	2000
4	p000000003	p000000005	2014-12-01 08:43:52	RYGC	50
5	p000000000	p000000004	2014-12-01 09:15:26	CB	200

MoneyInteractions:

Purpose:

This table is used to store transactions such as withdrawals and deposits of money into the stock exchange for the purpose of buying and selling stocks. While this may seem unnecessary at first, a typical stock market trading application utilizes liquid funds that are accumulated by the user either by depositing money into their account or selling stocks that they own. This stored monetary value can then be withdrawn from their account for general use. For this purpose, we track all monetary transactions by users so as to keep their accounts balanced, storing their unique ID, the date and time of the transaction and the amount of money deposited in US dollars (recording withdrawals as negative deposits).

SQL Create Statement:

```
CREATE TABLE MoneyInteractions (
  pID                 char(10) not null references People(pID),
  dateTime            timestamp not null,
  MoneyDepositedUSD   numeric(11,2) not null,
 primary key(pID, dateTime)
);
```

Functional Dependencies:

pID, dateTime → moneyDepositedUSD

Sample Data:

	pid character(10)	datetime timestamp without time zone	moneydepositedusd numeric(11,2)
1	p000000002	2014-11-15 00:00:01	50000.00
2	p000000004	2014-11-17 09:22:45	26.17
3	p000000001	2014-11-17 11:32:59	1250.50
4	p000000000	2014-11-19 16:08:31	450.00
5	p000000001	2014-11-25 04:15:19	-250.50

Industries:

Purpose:

This table records all of the industries which businesses on the stock market are involved in. This can be used to track all companies working in a certain industry so as to compare financial success and potential of a stock within a specific market or industry. To do so, this table stores a unique identifier for each industry represented on the stock exchange and a name for each industry.

SQL Create Statement:

```
CREATE TABLE Industries (
  industID            char(5) not null,
  industryName        text,
 primary key(industID)
);
```

Functional Dependencies:

industID → industryName

Sample Data:

	industid character(5)	industryname text
1	i0000	Food/Drink
2	i0001	Entertainment/Media
3	i0002	Ecological
4	i0003	Transportation
5	i0004	Educational
6	i0005	World Domination

MarketRelations:

Purpose:

This table is a relational table that relates companies on the stock exchange to the industry/industries in which they are involved so as to allow users of the database to query for all stocks in a specific industry (see view 'stocksByIndustry' in Views section of this report). To do so, this table stores the unique identifiers for the company and industry for each entry in this table, allowing for one company to be involved in many different industries and for one industry to have many different related companies/stocks on the stock exchange.

SQL Create Statement:

```
CREATE TABLE MarketRelations (
  cID                 text not null references Companies(cID),
  industID            char(5) not null references Industries(industID),
 primary key(cID, industID)
);
```

Functional Dependencies:

cID, industID →

Sample Data:

	cid text	industid character(5)
1	RCPR	i0000
2	APDR	i0000
3	CDR	i0000
4	CB	i0000
5	RR	i0003
6	MSFA	i0001
7	TBE	i0004
8	RYGC	i0002
9	RYGC	i0005

HistoricalStockPrices:

Purpose:

This table stores a history of all stock prices over an indefinite period of time determined by the users of the stock exchange system utilizing this database. For example, stock prices could be recorded in this table monthly so as to maintain a history of stock values which could be used for trend analysis and research by investors to determine potential future stock values and whether certain stocks appear to be increasing or decreasing in value. The functionality of this table depends upon the system that is inserting to this table and the time intervals over which inserts would occur, allowing this table to have a wide variety of functional uses by users of this database. The table stores the stock symbol for the stock in question, the month and year of the stock value as well as the stock's value in US dollars at the time of the table entry.

SQL Create Statement:

```
CREATE TABLE HistoricalStockPrices (
  cID                 text not null references Companies(cID),
  recordMonth         integer not null check (recordMonth in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)),
  recordYear          integer not null,
  stockPriceUSD       numeric(11,2) not null,
 primary key(cID, recordMonth, recordYear)
);
```

Functional Dependencies:

cID, recordMonth, recordYear → stockPriceUSD

Sample Data:

	cid text	recordmonth integer	recordyear integer	stockpriceusd numeric(11,2)
1	RCPR	8	2014	2.05
2	RCPR	9	2014	8.75
3	APDR	8	2014	4.13
4	APDR	9	2014	11.82
5	CDR	8	2014	4.26
6	CDR	9	2014	10.48
7	RR	8	2014	0.99
8	RR	9	2014	1.00
9	CB	8	2014	15.18
10	CB	9	2014	24.63
11	MSFA	8	2014	13.75
12	MSFA	9	2014	13.75
13	TBE	8	2014	51.03
14	TBE	9	2014	4.29
15	RYGC	8	2014	17.94
16	RYGC	9	2014	18.67

Brokerages:

Purpose:

This table stores all brokerages involved with trading on the Poughkeepsie Stock Exchange. Each brokerage firm can possibly have multiple portions of the firm that operate differently for different types of clients, so a firm may have multiple entries in this table, each acting firm entity having a distinct ID number in the table even if the brokerage name is the same.

SQL Create Statement:

```
CREATE TABLE Brokerages (
  bID                 char(10) not null,
  brokerageName       text not null,
 primary key(bID)
);
```

Functional Dependencies:

bID → brokerageName

Sample Data:

	bid character(10)	brokeragename text
1	b000000000	Plumber Bros.
2	b000000001	Wallace, Wallace & Wallace
3	b000000002	Your Money Is Our Money Inc.
4	b000000003	Responsible Money Handlers
5	b000000004	The Good Guys
6	b000000005	Green is Good

BrokerClientRelations:

Purpose:

This table stores all relations between brokerages and client on this stock exchange, functioning as a client list for each broker. This allows users of the database, in this case the brokers using the associated system, to maintain lists of all clients that they have as well as the fee in US dollars that they charge the client for each managed stock transaction.

SQL Create Statement:

```
CREATE TABLE BrokerClientRelations (
  bID                 char(10) not null references Brokerages(bID),
  pID                 char(10) not null references Investors(investorID),
  brokerTradeFeeUSD   numeric(8,2) not null default 0.00,
 primary key(bID, pID)
);
```

Functional Dependencies:

bID, pID → brokerTradeFeeUSD

Sample Data:

	bid character(10)	pid character(10)	brokertradefeeusd numeric(8,2)
1	b000000004	p000000001	4.25
2	b000000003	p000000004	3.15
3	b000000001	p000000007	6.18
4	b000000002	p000000003	10.75

StocksOwned:

Purpose:

This table stores all investors and the stocks that they own, including the number of each stock that each investor owns. This is useful for maintaining all records of stock ownership and calculating percent ownership of companies as well as overall stock value for each investor.

SQL Create Statement:

```
CREATE TABLE StocksOwned (
  investorID          char(10) not null references Investors(investorID),
  stock               text not null references Companies(cID),
  numberOfStocksOwned integer not null default 0,
 primary key(investorID, stock)
);
```

Functional Dependencies:

investorID, stock → numberOfStocksOwned

Sample Data:

	investorid character(10)	stock text	numberofstocksowned integer
1	p000000007	APDR	2684
2	p000000000	CB	2000
3	p000000002	CB	155
4	p000000002	CDR	125
5	p000000004	MSFA	2000
6	p000000008	RCPR	1725
7	p000000001	RR	320
8	p000000006	RR	80
9	p000000005	RYGC	726
10	p000000003	RYGC	123
11	p000000004	TBE	750

Views:

CompanyStockHolders:

Purpose:

The purpose of this view is to be able to see all investors in each company as well as the total number of stocks that each investor holds. This view sorts alphabetically by company name and secondarily by last name of each investor, so it becomes a very readable and useful table for users to research/analyze stock holdings throughout the market.

SQL Create Statement:

```
CREATE VIEW CompanyStockHolders AS
    SELECT c.cID AS StockSymbol,
           c.companyName AS Company,
           inv.investorID AS InvestorID,
           p.lastName AS LastName,
           p.firstName AS FirstName,
           so.numberOfStocksOwned AS StocksOwned
    FROM   Companies c,
           Investors inv,
           People p,
           StocksOwned so
    WHERE p.pID = inv.investorID
    AND   inv.investorID = so.investorID
    AND   c.cID = so.stock
    ORDER BY c.companyName ASC,
             p.lastName ASC;
```

Sample Output:

	stocksymbol text	company text	investorid character(10)	lastname text	firstname text	stocksowned integer
1	APDR	AlwaysPrepared Diner Restaurant	p000000007	Techsavvy	Zeke	2684
2	CDR	Castle Diner Restaurant	p000000002	Prime	Alan	125
3	CB	College Brewery	p000000000	Figs	Mike	2000
4	CB	College Brewery	p000000002	Prime	Alan	155
5	MSFA	MovieStreamingForAll	p000000004	Moneybags	Johnny	2000
6	RCPR	Random College Pizza Restaurant	p000000008	Hungryman	Harry	1725
7	RYGC	Recycle-Your-Goods Center	p000000005	Ecofriendly	Wendy	726
8	RYGC	Recycle-Your-Goods Center	p000000003	NewYear	Baby	123
9	RR	Riverside Railroad	p000000001	Roosevelt	Teddy	320
10	RR	Riverside Railroad	p000000006	Steel	Sandra	80
11	TBE	Textbook Emporium	p000000004	Moneybags	Johnny	750

BrokerClients:

Purpose:

This view is used as a listing of all customers of the brokerages involved on the Poughkeepsie Stock Exchange, useful for retrieving the relevant information about the brokers as well as their clients in one table. This could be used and adapted mainly for use by the stock brokers as a means of maintaining lists of their clients and possibly for finding potential future clients.

SQL Create Statement:

```
CREATE VIEW BrokerClients AS
    SELECT b.bID AS BrokerID,
           b.brokerageName AS BrokerageName,
           inv.investorID AS InvestorID,
           p.lastName AS ClientLastName,
           p.firstName AS ClinetFirstName
    FROM   Brokerages b,
           Investors inv,
           People p,
           BrokerClientRelations bcr
    WHERE  p.pID = inv.investorID
    AND    bcr.pID = inv.InvestorID
    AND    b.bID = bcr.bID
    ORDER BY b.brokerageName ASC,
             p.lastName ASC;
```

Sample Output:

	brokerid character(10)	brokeragename text	investorid character(10)	clientlastname text	clientfirstname text
1	b000000003	Responsible Money Handlers	p000000004	Moneybags	Johnny
2	b000000004	The Good Guys	p000000001	Roosevelt	Teddy
3	b000000001	Wallace, Wallace & Wallace	p000000007	Techsavvy	Zeke
4	b000000002	Your Money Is Our Money Inc.	p000000003	NewYear	Baby

StocksByIndustry:

Purpose:

This view displays the industries connected to this stock exchange as well as the companies involved in these industries and their corresponding stock prices. With all of this market data consolidated into one table without sensitive investor data, this view becomes very useful for market analysts seeking to observe the state of each industry as a whole as well as competition and market share within these industries as they relate to the Poughkeepsie Stock Exchange.

SQL Create Statement:

```
CREATE OR REPLACE VIEW StocksByIndustry AS
    SELECT i.industID AS IndustryID,
           i.industryName AS Industry,
           c.cID AS CompanyID,
           c.companyName AS CompanyName,
           c.stockPriceUSD AS StockValueUSD
    FROM   Industries i,
           Companies c,
           MarketRelations mr
    WHERE  mr.cID = c.cID
    AND    mr.industID = i.industID
    ORDER BY i.industryName ASC,
             c.companyName ASC;
```

Sample Output:

	industryid character(5)	industry text	companyid text	companyname text	stockvalueusd numeric(8,2)
1	i0002	Ecological	RYGC	Recycle-Your-Goods Center	11.11
2	i0004	Educational	TBE	Textbook Emporium	3.11
3	i0001	Entertainment/Media	MSFA	MovieStreamingForAll	16.25
4	i0000	Food/Drink	APDR	AlwaysPrepared Diner Restaurant	12.74
5	i0000	Food/Drink	CDR	Castle Diner Restaurant	13.99
6	i0000	Food/Drink	CB	College Brewery	20.45
7	i0000	Food/Drink	RCPR	Random College Pizza Restaurant	9.26
8	i0003	Transportation	RR	Riverside Railroad	2.13
9	i0005	World Domination	RYGC	Recycle-Your-Goods Center	11.11

Reports:

Calculating Sum of Client's Stock Values:

Purpose:

This query internally queries the StocksOwned, People and Companies tables to calculate the value of the individual stock holdings of each investor and then sums these values up to display the total stock value of each investor, along with their unique ID, first name and last name.

SQL Query:

```
SELECT inv.investorID, p.firstName, p.lastName, sumStockVals.sum AS TotalStockValueUSD
FROM   Investors inv,
       People p,
       (SELECT distinct id, sum(stockVal)
        FROM (SELECT p.PID as ID, (so.numberOfStocksOwned * c.stockPriceUSD) AS stockVal
              FROM   StocksOwned so,
                     Companies c,
                     People p
              WHERE  c.cID = so.stock
              AND    p.pID = so.investorID
             ) as stockValuations
        GROUP BY stockvaluations.id
       ) as sumStockVals
 WHERE inv.investorID = p.pID
 AND   sumStockVals.id = p.pID
 ORDER BY inv.investorID ASC;
```

Sample Output:

	investorid character(10)	firstname text	lastname text	totalstockvalueusd numeric
1	p000000000	Mike	Figs	40900.00
2	p000000001	Teddy	Roosevelt	681.60
3	p000000002	Alan	Prime	4918.50
4	p000000003	Baby	NewYear	1366.53
5	p000000004	Johnny	Moneybags	34832.50
6	p000000005	Wendy	Ecofriendly	8065.86
7	p000000006	Sandra	Steel	170.40
8	p000000007	Zeke	Techsavvy	34194.16
9	p000000008	Harry	Hungryman	15973.50

Stored Procedures:

AllStockOwners:

Purpose:

This procedure is designed to return information on all investors in a given stock on the PSE, including their name, ID number, the amount of the specified stock owned and the total amount of the specified stock that exists on the stock market. This function can be used by companies to keep track of all investors in their stocks and analyze the distribution of stocks amongst their investors.

SQL Create Statement:

```
CREATE OR REPLACE FUNCTION allStockOwners(text, refcursor) returns refcursor as
$$
DECLARE
    stockID    text    := $1;
    outputRef  refcursor := $2;
BEGIN
    OPEN outputRef for
        SELECT so.investorID,
               p.firstName,
               p.lastName,
               so.numberOfStocksOwned AS StocksOwned,
               c.totalNumberOfStocks AS StocksInMarket
        FROM   StocksOwned so,
               People p,
               Companies c
        WHERE  so.stock = stockID
        AND    p.pID = so.investorID
        AND    stockID = c.cID;
    RETURN outputRef;
END;
$$
language plpgsql;
```

AgeCheck:

Purpose:

This stored procedure is used as a helper function to the trigger ImplementAgeRestriction which checks the age of all people being entered into the database and confirms their age from their date of birth and the current date. If the client entered is not 18 years old or older, the database rejects the entry and returns a message to the user that the entry failed for age restrictions.

SQL Create Statement:

```
CREATE OR REPLACE FUNCTION ageCheck() RETURNS trigger AS
$$
DECLARE
BEGIN
    IF NEW.DOB > (now()::date - 6574) THEN
        DELETE FROM People
        WHERE People.pID = NEW.pID;
        RAISE NOTICE 'AGE RESTRICTION: Client entered is not 18 years of age or older, entry not saved';
    END IF;
    RETURN NEW;
END;
$$
language plpgsql;
```

Triggers:

ImplementAgeRestriction:

Purpose:

The purpose of this trigger, as explained above in the AgeCheck stored procedure section, is to prevent minors (younger than 18 years of age) from being entered into the database. There are legal regulations on the storage and handling of personal information of minors that complicates the maintenance of the database and minors are also not allowed to perform stock transactions or own stock directly. For these reasons, this trigger checks the date of birth of each client when an insert statement is called for them on the People table and rejects the insert if the client is younger than 18 years old.

SQL Create Statement:

```
CREATE TRIGGER implementAgeRestriction
AFTER INSERT ON People
FOR EACH ROW
EXECUTE PROCEDURE ageCheck();
```

Security:

Analyst User Role:

This user role is meant to allow market analysts to see the stock market data regarding companies, industries, stock prices over time and brokerages involved in stock market transactions. This user is only granted select access to the relevant tables so as to prevent analysts from altering or deleting any of the entries in the database.

```
CREATE USER analyst;
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM analyst;
GRANT SELECT ON Companies             TO analyst;
GRANT SELECT ON Industries            TO analyst;
GRANT SELECT ON MarketRelations       TO analyst;
GRANT SELECT ON HistoricalStockPrices TO analyst;
GRANT SELECT ON Brokerages            TO analyst;
```

TradeManager Auto User Role:

This user role is meant as an automated modifier of the Trades and StocksOwned tables when a trade occurs so as to automate the updating and transferring of funds and stocks between two clients. For this reason, the TradeManager role has select and insert access to these tables as well as update access to the StocksOwned table, allowing it to add rows and read the necessary data without altering previous trades in the Trade table.

```
CREATE USER TradeManager;
REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM analyst;
GRANT SELECT, INSERT ON Trades TO TradeManager;
GRANT SELECT, INSERT, UPDATE ON StocksOwned TO TradeManager;
```

DBAdmin User Role:

The database admin DBAdmin role has all privileges to the tables in the database as any good all-powerful DB admin should.

```
CREATE USER DBAdmin;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO DBAdmin;
```

Alan User Role:

Admin access for Alan, but don't try to add stocks or money to Alan Prime! It won't work…

```
CREATE USER alan;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO alan;
REVOKE INSERT, UPDATE ON StocksOwned       FROM alan;
REVOKE INSERT, UPDATE ON Trades            FROM alan;
REVOKE INSERT, UPDATE ON MoneyInteractions FROM alan;
```

Implementation Notes:

This database is designed to minimize the number of fields in each table so as to maximize the power of the relational database, but some inefficiencies may occur from the fact that some pieces of information are not able to be stored directly in tables in this design. For example, the sum monetary value of all of a client's stocks is not stored anywhere in a table and must be calculated any time that it is requested. This may be an issue if information of this form is requested frequently and by many users concurrently, possibly putting unnecessary strain on the system, but the decision to forego including a field for calculated values in the tables themselves was made due to the high frequency of stock price changes, meaning these fields would have to be updated far too frequently, thus reducing accuracy of data and/or efficiency.

In addition to this, please note that this database is only one portion of the full stock market system that would be required for a project of this scale. To incorporate a frontend application that allows users to interact with the database in an abstracted fashion, much more automation of SQL inserts, updates and triggers would need to be implemented so as to allow for values to change in any table and propagate the effects of these changes throughout all related tables in an efficient way that avoids manual input. For example, an automated insert of a client from the People table to the Investors table upon the client's first stock trade would need to be implemented so the admin would not need to manually add the user to the Investors table at this time. In future implementations of this system, considerations will have to be made for these automations and the owners/maintainers of the system would need to decide which are necessary and/or preferred for the system to run smoothly and effectively.

Known Problems:

- More automation/checks are needed to allow the database to update itself fully upon entry of data into certain tables. For example, upon an insert to the Trades table, checks need to be implemented to be sure that the buyer has enough funds in the system to buy the specified number of stocks, that the seller is in possession of these specified number of stocks before the trade, and automation is required to move the necessary funds and stocks between the two parties in question.
- Brokers currently have very little influence on trades and management of these trades. It is possible that another table is needed to specify trades that occur through a broker so as to accurately keep track of the fees that need to be charged to each client of the brokers and to subsequently take the appropriate amount of money out of their liquid funds in the system to transfer to the broker themselves.

- The HistoricalStockPrices table currently is implemented to store stock values on a minimum interval of once a month, but the recordMonth and recordYear fields should be consolidated into a timestamp field so as to allow for more frequent inserts of stock values into this table. This would make the analysis of stock prices over time much more useful for analysts as being able to see how a stock's price changes day-to-day or throughout one day of trading is very useful in an actual stock market.
- This database was originally designed as an in-use database, not as one that is started from an original state where no stocks are owned by investors and the companies themselves are in possession of the stocks corresponding to their company prior to opening of stock sales to the general public/investors. To accommodate for this, another entity would need to be created that abstracted investors and companies into 'traders' so as to allow for the initial sale of stocks from companies to investors to occur in the current Trades table.

Future Implementations:

In future versions of this database, I would like to:

- Implement the automation changes related to data effect propagation mentioned in the above 'Known Problems' section.
- Increase the role of brokers and insiders in the system and detect situations in which brokers have decisions/control over investors' portfolios and insiders have limitations on which other stocks they can invest in based upon their percentage ownership of companies in similar industries.
- Create an abstract 'client' user that is linked upon creation to a pID in the People table so as to allow for this user to only have access to stocks, trade history and broker information that is related to his/her own account and thus restrict this user from viewing the personal information of other users.
- Implement a detection system that monitors trading of specific stocks and alters the stock price based upon the trading activity and external factors in a similar fashion to real stock markets and their valuation of stocks in live markets.