The Users entity represents the user accounts and user information. Its attributes are UserID, Password, FullName, Address, and PhoneNumber. The UserID is the primary key and it is the user email that is unique to the user. This means UserID is type VARCHAR(255). Password is VARCHAR(255), FullName is VARCHAR(255), Address is VARCHAR(255), and PhoneNumber is VARCHAR(12) (e.g.123-456-7890). This table is its own entity to ensure that we maintain unique data for each user, as well as to reduce redundancy in the many-to-many relation with the Portfolios table.

Users:	
<u>UserID</u>	VARCHAR(255)
Password	VARCHAR(255)
FullName	VARCHAR(255)
Address	VARCHAR(255)
PhoneNumber	VARCHAR(12)

Users(UserID:VARCHAR(255) [PK], Password:VARCHAR(255), FullName:VARCHAR(255), Address:VARCHAR(255),
PhoneNumber:VARCHAR(12))

Portfolios is an entity that will consist of PortfolioID, PortfolioType, and PortfolioBalance. PortfolioID is the primary key and is type INT. Each portfolio will have its own unique identifier. PortfolioType is type VARCHAR(255) and is used to give a short description of the type of portfolio account or even just a name. PortfolioBalance is type DECIMAL(14,2) and keeps track of the cash value of that specific portfolio. Portfolio cannot be an attribute of Users as a user can have multiple portfolios so there would be multiple tuples with almost the same information except the one column. Thus, to remove the redundancy, Portfolios was made an entity. This is to replicate the real-life scenario where a person can open multiple accounts for various reasons such as a Roth, retirement, or individual account.

Portfolios:	
PortfolioID	INT
PortfolioType	VARCHAR(255)
PortfolioBalance	DECIMAL(14,2)

Portfolios(PortfolioID:INT [PK], PortfolioType:VARCHAR(255), PortfolioBalance:DECIMAL(14,2))

Users and Portfolio are connected using a junction table, UserPortfolio, as it is a many-to-many relationship. A user can have multiple portfolios and a portfolio can have multiple users that own it, making this a many-to-many relation. It would be a portfolio that multiple users collaborate on. A junction allows for easier linking between the two entities without having to overfill the entities with much redundant information. Its attributes are UserID and PortfolioID, which are the composite primary keys. Individually, they are foreign keys: UserID references Users and PortfolioID references Portfolios.

UserPortfolio	
UserID FK to Users	VARCHAR(255)
PortfolioID FK to Portfolios	INT

UsersPortfolio(UserID:VARCHAR(255) [PK] [FK to User.UserID], PortfolioID:INT [PK] [FK to Portfolios.PortfolioID])

Transactions is an entity that consists of all transaction information. Its attributes are TransactionID, PortfolioID, TransactionType, StockSymbol, NumShares, PurchasePrice, DateTime. TransactionID is the primary key with type INT. PortfolioID is a foreign key that references the Portfolios table, where PortfolioID is type INT. CurrentlyActive is type TINYINT(1), indicating the type of transaction that occurred and also indicates whether the information is viewable to the user. If the TransactionType is 0, the user is no longer able to view that transaction and means the shares were sold.

StockSymbol is type VARCHAR(10) and symbolizes which company stock was bought/sold from. NumShare is type INT, indicating the number of shares the user purchased/sold. It is type INT for the purpose of keeping this site as a simple introduction for people just starting out with investing. PurchasePrice is type DECIMAL(14,2) and indicates the price of the stock when the transaction occurs. This rounds off the value to the nearest hundredth. This information is collected through an API. DateTime is type TIMESTAMP and records when the transaction occurred as a log of the transaction. Transactions is an entity because each portfolio can have numerous transactions, so it would be redundant to have it as an attribute of Portfolios. Also, Transactions has its own attributes that we need to keep a record of. There are multiple transactions linked to a portfolio because the user can buy and sell multiple stocks however many times they want. A Transaction is linked to only one specific Portfolio as users can only buy/sell from within a specific portfolio, but a portfolio can have multiple transactions.

Transactions	
TransactionID	INT
PortfolioID FK to Portfolios	INT
CurrentlyActive	TINYINT(1)
StockSymbol	VARCHAR(10)
NumShares	INT
PurchasePrice	REAL
DateTime	TIMESTAMP

Transactions(TransactionID:INT [PK]: PortfolioID:INT [FK to Portfolios.PortfolioID], TransactionType:VARCHAR(4), StockSymbol:VARCHAR(10), NumShare:INT, PurchasePrice:REAL, DateTime:TIMESTAMP)

Watchlist is a junction table that joins the Portfolios and Companies tables allowing the user to look into the history of certain stocks and companies in each portfolio that the users want to observe. The attributes are StockSymbol and PortfolioID. Both are the primary keys and foreign keys: PortfolioID, type INT, references Portfolios. StockSymbol, type VARCHAR(10), references Companies. This watchlist will be limited to a specific amount of rows and act as “user favorited”. It has a many to many relation with Companies, as Portfolios will watch multiple companies and a company can be watched by multiple portfolios.

Watchlist	
StockSymbol FK to Companies	VARCHAR(10)
PortfolioID FK to Portfolios	INT

Watchlist(StockSymbol:VARCHAR(10) [PK] [FK to Companies.StockSymbol], PortfolioID:INT [PK] [FK to Portfolios.PortfolioID])

The Companies table stores general information about the companies that the user is interested in looking into. The attributes are StockSymbol as VARCHAR(10), Name as VARCHAR(255), Sector as VARCHAR(255), Industry as VARCHAR(255), MarketCap as DECIMAL(14,2), RevenueGrowth as DECIMAL(14,2), City as VARCHAR(255), State as VARCHAR(255), and Country as VARCHAR(255). StockSymbol is the primary key. Name is the long name of the company, sector and industry is what sector and industry the company lies in. For example, Apple Inc. is in the Technology sector and its industry is Consumer Electronics. MarketCap and RevenueGrowth are the market cap and revenue growth, respectively, of that company. City, State, and Country are where the Company is located. Its relation to Portfolios is many to many as the same stock can be “watched” in multiple different portfolios and each Portfolio can “watch” multiple stocks.

Companies	

StockSymbol	VARCHAR(10)
Name	VARCHAR(255)
Sector	VARCHAR(255)
Industry	VARCHAR(255)
MarketCap	DECIMAL(14,2)
RevenueGrowth	DECIMAL(14,2)
City	VARCHAR(255)
State	VARCHAR(255)
Country	VARCHAR(255)

Companies(StockSymbol:VARCHAR(10) [PK], Name:VARCHAR(255), Sector:VARCHAR(255), Industry:VARCHAR(255), MarketCap:DECIMAL(14,2), RevenueGrowth:DECIMAL(14,2), City:VARCHAR(255), State:VARCHAR(255), Country:VARCHAR(255))

…

HistoricalStocks is a table that consists of the close price of the stock everyday since 2010 till 2024. The attributes are StockSymbol as VARCHAR(10), Date as DATE, and ClosePrice as DECIMAL(14,2). The StockSymbol and Date are composite keys, giving each row a unique identifier. Stocks was created as a separate entity because the same stock symbol is present numerous times in this database, so if the attributes in Stocks were attributes of Companies, the same information would also be repeated numerous times. Splitting it allows the tables to be much cleaner and be less redundant with the information. The relation to Companies is one to many because each company has multiple stock information, since there is information for each day since 2015, and each stock info has only one company. We plan to use this table to find information such as the historical highs and/or lows and make it viewable to the user.

HistoricalStocks	
StockSymbol FK to Portfolios	VARCHAR(10)
Date	DATE
ClosePrice	DECIMAL(14,2)

Stocks(StockSymbol:VARCHAR(10) [PK], Date:TIMESTAMP, ClosePrice:REAL)

Cardinality Explanations:

Users <--> Portfolios:

Multiple Users can collaborate on one Portfolio, and one User can have multiple Portfolios, making this a many-to-many relationship.

Portfolios <--> Transaction:

A portfolio can have zero to many transactions but each transaction is linked to only one portfolio, making this a one-to-many relation.

Portfolios <--> Companies:

A portfolio can have zero to many companies on its watchlist and a company can be on zero to many portfolio watchlists, making this a many-to-many relation.

Companies <--> HistoricalStocks:

A company can have zero to many stocks but a stock can only be owned by one company, making this a one-to-many relation.