# Stock transaction processor

This code reads stock prices since the beginning of the experiment and transactions and shows each player's total.

## The Function

We will create a function that takes as arguments file names of players, stocks, and transactions and returns a table with the players sorted in order of their portfolio performance.

We start by calling the function and displaying the resulting table.

### Test Data

While the function will eventually support 101 stocks, dozens of players, and hundreds of transactions, we start with a small list of players, stocks and transactions.

**`TestPlayers.csv`**

This file contains three players. Normally we use email addresses for player names:

```
"email"

"Player1"

"Player2"

"Player3"
```

**`TestStocks.csv`**

This file contains the list of stocks that we'll use for all transactions.

```
AMZN

CSCO

META

NVDA
```

**`TestTransactions.csv`**

The transactions have four field in a CSV file

Timestamp,Email Address,Ticker,Type,Shares

```
3/4/2024 18:42:51,Player1,NVDA,Buy,11

3/6/2024 18:42:51,Player1,NVDA,Sell,11

3/7/2024 18:42:51,Player1,NVDA,Sell,11

3/8/2024 18:42:51,Player1,NVDA,Buy,50

3/5/2024 00:00:00,Player3,CSCO,Buy, 2

3/9/2024 18:42:51,Player1,NVDA,Buy,8
```

```
3/9/2024 18:42:51,Player2,NVDA,Buy,10

3/10/2024 18:42:51,Player3,NVDA,Buy,10

3/9/2024 18:42:51,Player1,AMZN,Buy,7

3/9/2024 18:42:51,Player2,AMZN,Buy,2

3/10/2024 18:42:51,Player3,AMZN, Buy,7

3/11/2024 00:00:00, notAPlayer,AMZN, Sell, 10

3/11/2024 00:00:00, Player3,AMZN, Sell, 5

3/19/2024 18:42:51,Player2,AMZN,Sell,1

3/19/2024 00:00:00, Player2, META, Sell, 7

3/20/2024 00:00:00, Player3, AMZN, Sell, 10
```

We call `processTransactions()` using the three files.

The `processTransactions()` function prints out the transactions and error messages. Then it returns the table with the players sorted in descending order of their portfolio value.

```
TRAN: Player1 wants to Buy 11.000 shares of NVDA for $9375.63 total
TRAN: Player1 wants to Sell 11.000 shares of NVDA for $9757.00 total
TRAN: Player1 wants to Sell 11.000 shares of NVDA for $10193.59 total
Player1 has 0.00 shares of NVDA and wants to sell 11.00
TRAN: Player1 wants to Buy 50.000 shares of NVDA for $43764.00 total
Player1 has 10381.37 dollars and needs 43764.00 to buy 50.00 shares of NVDA
TRAN: Player3 wants to Buy 2.000 shares of CSCO for $97.11 total
TRAN: Player1 wants to Buy 8.000 shares of NVDA for $7002.24 total
TRAN: Player2 wants to Buy 10.000 shares of NVDA for $8752.80 total
TRAN: Player3 wants to Buy 10.000 shares of NVDA for $8752.80 total
TRAN: Player1 wants to Buy 7.000 shares of AMZN for $1227.45 total
TRAN: Player2 wants to Buy 2.000 shares of AMZN for $350.70 total
TRAN: Player3 wants to Buy 7.000 shares of AMZN for $1227.45 total
Player3 has 1150.09 dollars and needs 1227.45 to buy 7.00 shares of AMZN
TRAN: notAPlayer wants to Sell 10.000 shares of AMZN for $1719.60 total
notAPlayer is not in the player list
TRAN: Player3 wants to Sell 5.000 shares of AMZN for $859.80 total
Player3 has 0.00 shares of AMZN and wants to sell 5.00
TRAN: Player2 wants to Sell 1.000 shares of AMZN for $175.90 total
TRAN: Player2 wants to Sell 7.000 shares of META for $3473.68 total
Player2 has 0.00 shares of META and wants to sell 7.00
TRAN: Player3 wants to Sell 10.000 shares of AMZN for $1781.50 total
Player3 has 0.00 shares of AMZN and wants to sell 10.00
```

`finalTable = 3x2 table`

| | players | CashValue |
|---|---|---|
| 1 | Player1 | 10509.47 |
| 2 | Player2 | 10077.13 |
| 3 | Player3 | 10065.67 |

## The processTransactions() function

Now we define the function.  It takes three file names as arguments and returns a table.

### Read in the data

We store data that remains the same across many transactions as catagorical.  This includes player names and stock ticker symbols.

### Set up player data

As players buy shares of stock, we store their stock totals in a matrix.  One row per player and one column per stock.

Everyone gets $10,000 to start.

### Read the stock history

We read all the stock price history for the stocks in the stock list from the beginning of the year to today.  We use this data to determine profit, loss, and final value of each stock.

### Read the transactions

We used the import tool to create code that sets up the transaction table. The `opts` variable holds all the import options.

The readtimetable() function reads transactions.  The function assumes that we have date/time data in the first column, which we do.

### Clear the time of day data.

We don't care what time of day the transctions happen as we only look at stock prices at the end of day.  The dateshift() function moves all the datetimes to the start of the day (00:00:00)

## Processing Transactions

Now we'll loop through the transactions in the table row-by-row and read their data.

### Execute the transaction

Now we have all the transaction data.  We change the stock holdings and cash to match the transaction instructions.

First find the player's row in the cash and holdings matrices.

Now find the column for the stock in the stocks array.

#### Are we buying or selling

If we are buying, check that we have enough cash to make the purchase.  Then subtract the cash from the `cash` vector and add the stock holdings to the `holdings` array.

If we are selling, check that the player has enough shares in the holdings array to make the sale.  Then subtract the sale from `holdings` and add the player's cash to `cash`.

## Output each player's final value

We have processed all the transactions.  It is time to find the cash value of the holdings using the most recent `adjClose` prices.