

# Pair Trading Project

- Design, populate, and query a sqlite3 database for pairs trading.
- Provide SQL queries for pair trading statistics.
- Write an Object-Oriented program in C++ to run the back-test for a given portfolio over a given time period and store the results in the database.
- Design the algorithm to implement real time “grey box” trading using the database and objects you used for the back tester.

# Condition and Assumption

- “Going short” – the first stock of the pair is short and the other is long.
- “Going long” - the first stock of the pair is long and the other is short.
- Always trade 10,000 shares for the first stock ( $S_1$ ) and determine the shares for the other ( $S_2$ ) accordingly:  $N_1P_1 + N_2P_2 = 0$
- $N_1$  and  $N_2$  are the numbers of shares of  $S_1$  and  $S_2$ , and  $P_1$  and  $P_2$  are the prices of  $S_1$  and  $S_2$

# Pair Trading Algorithm (1/2)

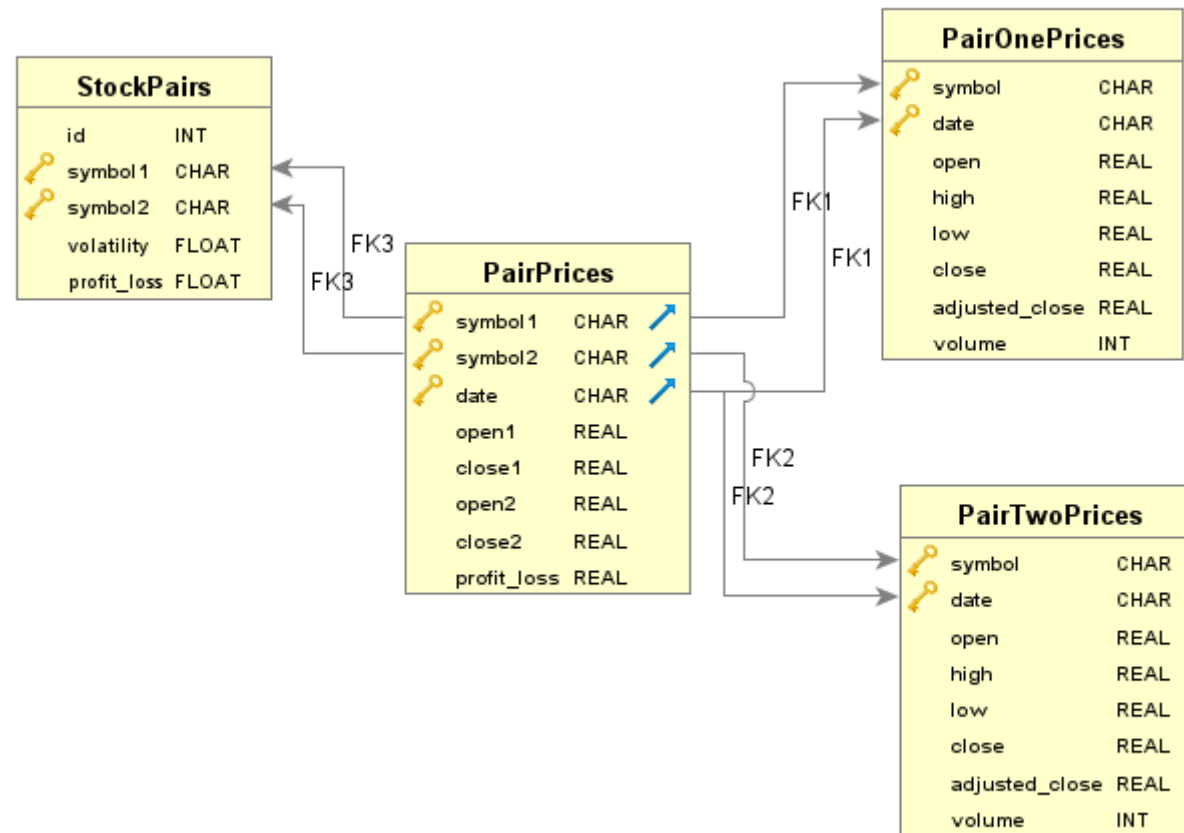
- Compute the standard deviation,  $\sigma_p$ , of the ratio of the two adjusted closing stock prices in each pair from 1/1/2011 to 12/31/2020. Store this standard deviation in the database.
- The variable,  $k$ , has default value 1, but could be changed by user in the pair trading program.
- Get  $Close1d1$ ,  $Close2d1$ ,  $Open1d2$ ,  $Open2d2$ ,  $Close1d2$ , and  $Close2d2$ , where  $Close1d1$  and  $Close2d1$  are the closing prices on day  $d - 1$  for stocks 1 and 2, respectively,  $Open1d2$ ,  $Open2d2$  are the opening prices for day  $d$ .

# Pair Trading Algorithm (2/2)

- Open Trade:
  - If  $\text{abs}(\text{Close1d1}/\text{Close2d1} - \text{Open1d2}/\text{Open2d2}) > k\sigma$ ,
    - short the pair;
  - Else if  $\text{abs}(\text{Close1d1}/\text{Close2d1} - \text{Open1d2}/\text{Open2d2}) < k\sigma$ ,
    - go long the pair.
  - $N1 = 10,000$  shares, traded at the price  $\text{Open1d2}$ ,
  - $N2 = N1 * (\text{Open1d2}/\text{Open2d2})$ , traded at the price  $\text{Open2d2}$ .
- Close Trade:
  - The open trades will be closed at the closing prices and P/L for the pair trade will be calculated as:
$$(\pm N1 * [\text{Open1d2} - \text{Close1d2}]) +$$
$$(\pm N2 * [\text{Open2d2} - \text{Close2d2}])$$

# Database Implementation Details (1)

- The database PairTrading.db has 4 tables: StockPairs, PairOnePrices, PairTwoPrices, PairPrices. They all have composite primary keys.



# Database Implementation Details (2)

- Create StockPairs table by using SQL DDL statement
- Populate the table with SQL DML statement
- Symbol1 and Symbol2 are read from PairTrading.txt

Table: StockPairs

	id	symbol1	symbol2	volatility	profit_loss
	Filter	Filter	Filter	Filter	Filter
1	1	AAPL	HPQ	0.0	0.0
2	2	AXP	COF	0.0	0.0
3	3	BAC	JPM	0.0	0.0
4	4	BHP	FCX	0.0	0.0
5	5	CAT	DE	0.0	0.0
6	6	CHK	DVN	0.0	0.0
7	7	COP	CVX	0.0	0.0
8	8	CS	DB	0.0	0.0
9	9	CSX	NSC	0.0	0.0
10	10	CVX	XOM	0.0	0.0
11	11	DAL	UAL	0.0	0.0
12	12	DSX	GNK	0.0	0.0
13	13	EGLE	GNK	0.0	0.0
14	14	HD	LOW	0.0	0.0
15	15	IEF	TLT	0.0	0.0
16	16	INTC	SMH	0.0	0.0
17	17	KGC	NEM	0.0	0.0
18	18	T	VZ	0.0	0.0
19	19	UGA	USO	0.0	0.0

# Database Implementation Details (3)

- PairOnePrices and PairTwoPrices tables are created with DDL statement.
- Using libcurl to pull daily historical data from 2011/1/1 to 2020/12/31 for each stock in the pairs
- Using json parser to parse the daily historical data
- Using DML Insert statement to populate the tables

# Database Implementation Details (4)

Table: PairOnePrices





	symbol	date	open	high	low	close	adjusted_close	volume
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
197	AAPL	2020-03-12	255.94	270.0	248.0	248.23	248.23	104618500
198	AAPL	2020-03-13	264.89	279.92	252.95	277.97	277.97	92683000
199	AAPL	2020-03-16	241.95	259.08	240.0	242.21	242.21	80605900
200	AAPL	2020-03-17	247.51	257.61	238.4	252.86	252.86	81014000
201	AAPL	2020-03-18	239.77	250.0	237.12	246.67	246.67	75058400
202	AAPL	2020-03-19	247.39	252.84	242.61	244.78	244.78	67964300
203	AAPL	2020-03-20	247.18	251.83	228.0	229.24	229.24	100257000
204	AAPL	2020-03-23	228.08	228.5	212.61	224.37	224.37	84188200
205	AAPL	2020-03-24	236.36	247.69	234.3	246.88	246.88	71882800
206	AAPL	2020-03-25	250.75	258.25	244.3	245.52	245.52	75900500
207	AAPL	2020-03-26	246.52	258.68	246.36	258.44	258.44	63021800
208	AAPL	2020-03-27	252.75	255.87	247.05	247.74	247.74	51007500
209	AAPL	2020-03-30	250.74	255.52	249.4	254.81	254.81	41994100
210	AAPL	2020-03-31	255.6	262.49	252.0	254.29	254.29	49250500
211	AAPL	2020-04-01	246.5	248.72	239.13	240.91	240.91	44054600
212	AAPL	2020-04-02	240.34	245.15	236.9	244.93	244.93	41483500
213	AAPL	2020-04-03	242.8	245.7	238.97	241.41	241.41	32418200
214	AAPL	2020-04-06	250.9	263.11	249.38	262.47	262.47	50455100
215	AAPL	2020-04-07	270.8	271.7	259.0	259.43	259.43	50646300
216	AAPL	2020-04-08	262.74	267.37	261.23	266.07	266.07	39582531



# Database Implementation Details (5)

- PairPrices table is created by using DDL statement, including both primary keys and foreign keys.
- The table is populated by querying other 3 tables:
- Insert into PairPrices
  - Select     StockPairs.symbol1 as symbol1, StockPairs.symbol2 as symbol2,  
                    PairOnePrices.date as date,  
                    PairOnePrices.open as open1, PairOnePrices.close as close1,  
                    PairTwoPrices.open as open2, PairTwoPrices.close as close2,  
                    0 as profit\_loss
  - From StockPairs, PairOnePrices, PairTwoPrices
  - Where (((StockPairs.symbol1 = PairOnePrices.symbol) and  
                    (StockPairs.symbol2 = PairTwoPrices.symbol)) and  
                    (PairOnePrices.date = PairTwoPrices.date))
  - ORDER BY symbol1, symbol2;

# Database Implementation Details (6)

Table: PairPrices     New R

	symbol1	symbol2	date	open1	close1	open2	close2	profit_loss
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
197	AAPL	HPQ	2020-03-12	255.94	248.23	17.4	17.51	0.0
198	AAPL	HPQ	2020-03-13	264.89	277.97	18.57	17.18	0.0
199	AAPL	HPQ	2020-03-16	241.95	242.21	15.72	14.46	0.0
200	AAPL	HPQ	2020-03-17	247.51	252.86	14.64	14.4	0.0
201	AAPL	HPQ	2020-03-18	239.77	246.67	13.55	13.11	0.0
202	AAPL	HPQ	2020-03-19	247.39	244.78	13.01	14.4	0.0
203	AAPL	HPQ	2020-03-20	247.18	229.24	14.42	13.95	0.0
204	AAPL	HPQ	2020-03-23	228.08	224.37	13.78	13.7	0.0
205	AAPL	HPQ	2020-03-24	236.36	246.88	14.29	15.47	0.0
206	AAPL	HPQ	2020-03-25	250.75	245.52	15.48	15.38	0.0
207	AAPL	HPQ	2020-03-26	246.52	258.44	15.53	17.94	0.0
208	AAPL	HPQ	2020-03-27	252.75	247.74	17.0	16.86	0.0
209	AAPL	HPQ	2020-03-30	250.74	254.81	16.93	17.84	0.0
210	AAPL	HPQ	2020-03-31	255.6	254.29	17.84	17.36	0.0
211	AAPL	HPQ	2020-04-01	246.5	240.91	16.25	14.84	0.0
212	AAPL	HPQ	2020-04-02	240.34	244.93	14.81	15.49	0.0
213	AAPL	HPQ	2020-04-03	242.8	241.41	15.39	14.48	0.0
214	AAPL	HPQ	2020-04-06	250.9	262.47	15.22	15.0	0.0
215	AAPL	HPQ	2020-04-07	270.8	259.43	15.7	15.12	0.0
216	AAPL	HPQ	2020-04-08	262.74	266.07	15.44	15.72	0.0

# Database Implementation Details (7)

- Calculate variance for each pair and update the volatility for each pair:
  - `string calculate_volatility_for_pair = string("Update StockPairs SET volatility =")`
  - `+ "(SELECT(AVG((Close1/Close2)*(Close1/Close2)) - AVG(Close1/Close2)*AVG(Close1/Close2)) as variance "`
  - `+ "FROM PairPrices "`
  - `+ "WHERE StockPairs.symbol1 = PairPrices.symbol1 AND StockPairs.symbol2 = PairPrices.symbol2 AND PairPrices.date <= \"`
  - `+ back_test_start_date + "\");";`
- Data range for volatility: 1/1/2011 – 12/31/2020

# Database Implementation Details (8)

Table: StockPairs

	id	symbol1	symbol2	volatility	profit_loss
	Filter	Filter	Filter	Filter	Filter
1	1	AAPL	HPQ	3.0580092733...	0.0
2	2	AXP	COF	0.0061376161...	0.0
3	3	BAC	JPM	1.6880666779...	0.0
4	4	BHP	FCX	0.1264185056...	0.0
5	5	CAT	DE	0.0012595325...	0.0
6	6	CHK	DVN	0.0002365275...	0.0
7	7	COP	CVX	0.0006241256...	0.0
8	8	CS	DB	0.0075423339...	0.0
9	9	CSX	NSC	0.0001029702...	0.0
10	10	CVX	XOM	0.0015340749...	0.0
11	11	DAL	UAL	0.0007584272...	0.0
12	12	DSX	GNK	0.0010687526...	0.0
13	13	EGLE	GNK	0.0053282721...	0.0
14	14	HD	LOW	0.0101586352...	0.0
15	15	IEF	TLT	0.0003352295...	0.0
16	16	INTC	SMH	0.0001053624...	0.0
17	17	KGC	NEM	0.0001120383...	0.0
18	18	T	VZ	0.0007173429...	0.0
19	19	UGA	USO	0.0059604787...	0.0

The volatility values are for examples, your values will be different

# Database Implementation Details (9)

- Back Te

Table: PairPrices

	symbol1	symbol2	date	open1	close1	open2	close2	profit_loss
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
148	AAPL	HPQ	2019-12-31	289.93	293.65	20.41	20.55	0.0
149	AAPL	HPQ	2020-01-02	296.24	300.35	20.68	20.79	25342.61
150	AAPL	HPQ	2020-01-03	297.15	297.43	20.48	20.53	-4454.6
151	AAPL	HPQ	2020-01-06	293.79	299.8	20.66	20.61	67210.1
152	AAPL	HPQ	2020-01-07	299.84	298.39	20.56	20.66	-29083.6
153	AAPL	HPQ	2020-01-08	297.16	303.19	20.53	20.93	2402.4
154	AAPL	HPQ	2020-01-09	307.24	309.63	21.0	21.24	-11212.96
155	AAPL	HPQ	2020-01-10	310.6	310.33	21.26	21.38	-20231.4
156	AAPL	HPQ	2020-01-13	311.64	316.96	21.46	21.46	53200.0
157	AAPL	HPQ	2020-01-14	316.7	312.68	21.51	21.4	-24004.37
158	AAPL	HPQ	2020-01-15	311.85	311.34	21.35	21.42	-15324.55
159	AAPL	HPQ	2020-01-16	313.59	315.24	21.41	21.67	-21581.68
160	AAPL	HPQ	2020-01-17	316.27	318.73	21.63	21.97	-25114.12
161	AAPL	HPQ	2020-01-21	317.19	316.57	21.92	21.96	-11988.12

The values in profit\_loss is for demo only. Your values will be different

# Database Implementation Details (10)

Populate profit\_loss for each pair in StockPairs table:

StockPairs				
id	symbol1	symbol2	volatility	profit_loss
Filter	Filter	Filter	Filter	Filter
1	AAPL	HPQ	3.0580092733...	163935.58
2	AXP	COF	0.0061376161...	227171.24
3	BAC	JPM	1.6880666779...	3064.91
4	BHP	FCX	0.1264185056...	80548.69
5	CAT	DE	0.0012595325...	90993.68
6	CHK	DVN	0.0002365275...	-4214.65
7	COP	CVX	0.0006241256...	-134014.17
8	CS	DB	0.0075423339...	22743.54
9	CSX	NSC	0.0001029702...	12322.84
10	CVX	XOM	0.0015340749...	93443.75
11	DAL	UAL	0.0007584272...	136301.5
12	DSX	GNK	0.0010687526...	-4841.11
13	EGLE	GNK	0.0053282721...	-9343.3
14	HD	LOW	0.0101586352...	480589.27
15	IEF	TLT	0.0003352295...	219637.86
16	INTC	SMH	0.0001053624...	116457.39
17	KGC	NEM	0.0001120383...	1412.07
18	T	VZ	0.0007173429...	-26287.41
19	UGA	USO	0.0059604787...	2532.61

The values in profit\_loss is for demo only. Your values will be different

# Pair Trading C++ Classes (1)

```
class TradeData
{
private:
    string sDate;
    double dOpen;
    double dHigh;
    double dLow;
    double dClose;
    double dAdjClose;
    long lVolume;
public:
    TradeData() : sDate(""), dOpen(0), dClose(0), dHigh(0),
    TradeData(string sDate_, double dOpen_, double dHigh_,
        sDate(sDate_), dOpen(dOpen_), dHigh(dHigh_), dLow(0), dClose(0), dAdjClose(0), lVolume(0))
    TradeData(const TradeData & TradeData) : sDate(TradeData.sDate), dOpen(TradeData.dOpen), dHigh(TradeData.dHigh), dLow(TradeData.dLow), dClose(TradeData.dClose), dAdjClose(TradeData.dAdjClose), lVolume(TradeData.lVolume)
    TradeData operator=(const TradeData & TradeData)
    {
        sDate = TradeData.sDate;
        dOpen = TradeData.dOpen;
        dHigh = TradeData.dHigh;
        dLow = TradeData.dLow;
        dClose = TradeData.dClose;
        dAdjClose = TradeData.dAdjClose;
        lVolume = TradeData.lVolume;

        return *this;
    }
}
```

# Pair Trading C++ Classes (2)

```
class Stock
{
private:
    string sSymbol;
    vector<TradeData> trades;

public:
    Stock() : sSymbol("") {}
    Stock(string sSymbol_, const vector<TradeData> trades_) :sSymbol(sSymbol_), trades(trades_) {}
    Stock(const Stock & stock) :sSymbol(stock.sSymbol), trades(stock.trades) {}
    Stock operator=(const Stock & stock)
    {
        sSymbol = stock.sSymbol;
        trades = stock.trades;

        return *this;
    }

    void addTrade(const TradeData & trade) { trades.push_back(trade); }
    string getSymbol() { return sSymbol; }
    const vector<TradeData> & getTrades() const { return trades; }

    friend ostream & operator<<(ostream & ostr, const Stock & stock)
    {
        ostr << "Symbol: " << stock.sSymbol << endl;
        for (vector<TradeData>::const_iterator itr = stock.trades.begin(); itr != stock.trades.end(); itr++)
            ostr << *itr;
        return ostr;
    }
};
```



# Pair Trading C++ Classes (3)

```
struct PairPrice
{
    double dOpen1;
    double dClose1;
    double dOpen2;
    double dClose2;
    double dProfit_Loss;
    PairPrice() : dOpen1(0), dClose1(0), dOpen2(0), dClose2(0), dProfit_Loss(0)
    {
    }
    PairPrice(double dOpen1_, double dClose1_, double dOpen2_, double dClose2_) :dOpen1(dOpen1_), dClose1(dC]
    {
    }
};
```

# Pair Trading C++ Classes (4)

```
class StockPairPrices
{
private:
    pair<string, string> stockPair;
    double volatility;
    double k;
    map<string, PairPrice> dailyPairPrices;
public:
    StockPairPrices() { volatility = 0; k = 0; }
    StockPairPrices(pair<string, string> stockPair_) { stockPair = stockPair_; volatility = 0; k = 0; }
    void SetDailyPairPrice(string sDate_, PairPrice pairPrice_)
    {
        dailyPairPrices.insert(pair<string, PairPrice>(sDate_, pairPrice_));
    }
    void SetVolatility(double volatility_) { volatility = volatility_; }
    void SetK(double k_) { k = k_; }
    void UpdateProfitLoss(string sDate_, double dProfitLoss_)
    {
        dailyPairPrices[sDate_].dProfit_Loss = dProfitLoss_;
    }
    pair<string, string> GetStockPair() const
    {
        return stockPair;
    }
    map<string, PairPrice> GetDailyPrices() const
    {
        return dailyPairPrices;
    }
    double GetVolatility() const
    {
        return volatility;
    }
    double GetK() const
    {
        return k;
    }
}
```

# Pair Trading Program Menu

```
Menu
=====
A - Create and Populate Pair Table
B - Retrieve and Populate Historical Data for Each Stock
C - Create PairPrices Table
D - Calculate Volatility
E - Back Test
F - Calculate Profit and Loss for Each Pair
G - Manual Testing
H - Drop All the Tables
X - Exit

Enter selection:
```

# References

- *Database systems: The Complete Book*. Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom. Prentice Hall, 2008, ISBN 0-131-87325-3
- *Database Fundamentals*, Robert Robbins, John Hopkins University, 1995
- *Relational Database*, INFM 603 Week 9, College of Information Studies, University of Maryland, 2014
- *NYU Polytechnic School, FRE6883, Financial Computing, Lecture Notes, Fall 2014*
- *Pairs Trading, Quantitative Methods and Analysis*, Ganapathy Vidyamurthy, Wiley, 2004, ISBN 0-471-46067-2
- *Pairs Trading: A Bayesian Example*, Stefan Hools & J.Richard Hollos, Abrazol Publishing, 2012
- *High-Frequency Trading: A Practical Guide to Algorithmic Strategies & Trading Systems*, 2nd Ed, 2013, Irene Aldridge, ISBN: 1-118-34350-6