Loading Libraries and importing files

In [50]:
```
# Importing files and loading libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches
import matplotlib.dates as mdates
import os
from matplotlib.pyplot import figure
import warnings
warnings.filterwarnings('ignore')
```

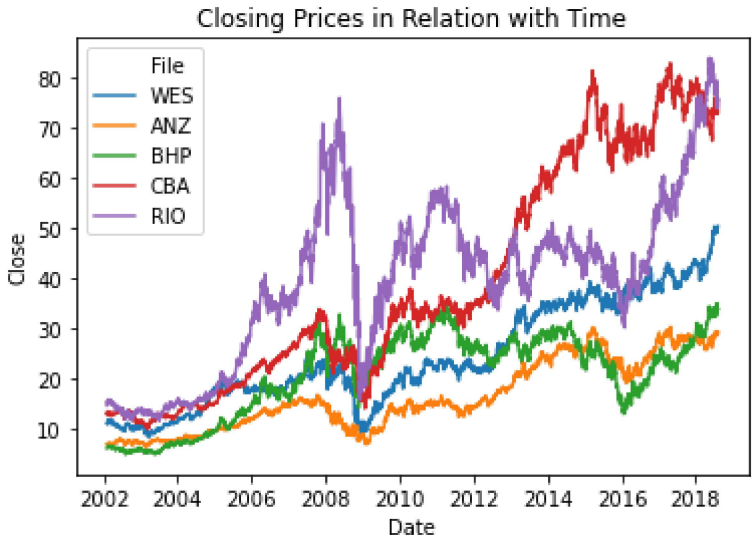
In [51]:
```
# Function to read all CSV files from directory and return them in one data
frame
def read_dir(dir):
    files = pd.concat((pd.read_csv(dir + fname).assign(File = fname) for fn
ame in os.listdir(dir)))
    files["File"] = [file.replace(".csv", "") for file in files["File"]]
    return(files)

# Utilized code from: Topic 9 lab questions
```

In [52]:
```
# Loading in stocks
stocks = read_dir('/Users/siam/Desktop/Coding Assessment/Given_Files/Stock
s/')
```

Task 1

In [53]:
```
# Converting 'Date' to date-time object
stocks.Date = pd.to_datetime(stocks.Date)
```

In [55]:
```
# Creating line plot that showcases the relationship between closing prices
and time
filtered_stocks = stocks[stocks.File.isin(["ANZ", "BHP", "CBA", "RIO", "WE
S"])]

sns.lineplot(x = "Date", y = "Close", hue = "File", data = filtered_stocks)
plt.title("Closing Prices in Relation with Time")
plt.show()

# Code utilized from: Lab Questions Topic 9 Task 5
```



In [56]:
```
# Finding correlation of closing stock prices
close_corr = stocks[['Date', 'Close', 'File']]

close_corr_pivot = close_corr.pivot('Date','File','Close').reset_index()

close_corr = close_corr_pivot.corr(method='pearson')

close_corr.head(10)

# Code utilized from: https://www.interviewqs.com/blog/py_stock_correlation
```

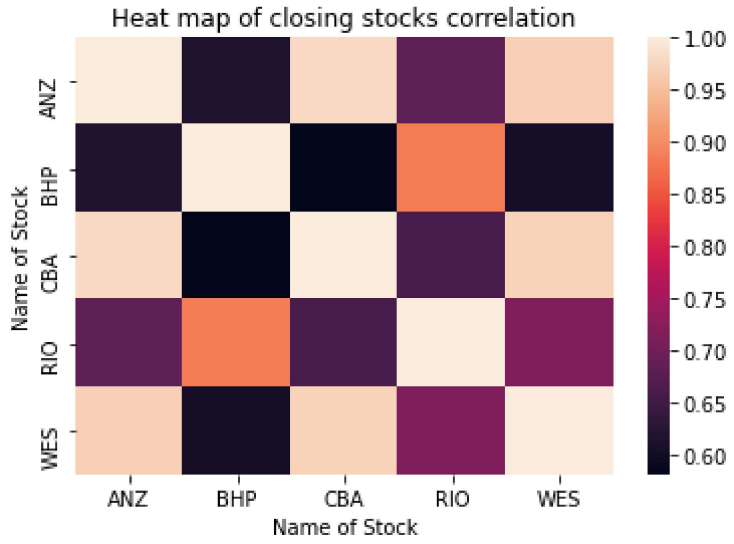
Out[56]:

File	ANZ	BHP	CBA	RIO	WES
File					
ANZ	1.000000	0.614569	0.980258	0.683818	0.969762
BHP	0.614569	1.000000	0.581494	0.884080	0.604126
CBA	0.980258	0.581494	1.000000	0.660790	0.973428
RIO	0.683818	0.884080	0.660790	1.000000	0.716878
WES	0.969762	0.604126	0.973428	0.716878	1.000000

The table above is a representation of the correlation between the stocks, ANZ, BHP, CBA, RIO and WES. The stock correlation identifies the relationship between these stocks and the price movementsm it is evident that the column 'ANZ' and the row 'ANZ' represent a positive 1 which is a rare occurance of perfect positive correlation, including the columns and rows, 'BHP', 'CBA', 'RIO', and 'WES'.

The remaining data in the table above are between +1 and -1. Therefore, the investor should choose assets that have low correlation with each other in order to reduce the risk within the portfolio.

In [57]:
```
# Heat map of 'stocks.corr'
sns.heatmap(close_corr)
plt.title("Heat map of closing stocks correlation")
plt.xlabel("Name of Stock")
plt.ylabel("Name of Stock")
plt.show()

# Utilized code from: https://seaborn.pydata.org/generated/seaborn.heatmap.html
```



The heatmap graph above representing the closing correlation between the stocks illustrates that the rare occurances of the perfect positive correlations is a diagonal line from the top towards the bottom. Furthermore, the graph showcases the remaining data points of the correlation of the stocks.

Task 2

Moving Averages

In [58]:
```
# Finding moving average of 5 and 20
filtered_stocks['MAV5'] = filtered_stocks['Open'].rolling(window=5).mean()
filtered_stocks['MAV20'] = filtered_stocks['Open'].rolling(window=20).mean()

# Code utilized from: https://www.learndatasci.com/tutorials/python-finance-part-3-moving-average-trading-strategy/
```

Last 5 values of each stock

In [59]:
```
# Sorting each stock into a seperate dataframe
wes_t5 = filtered_stocks[filtered_stocks['File'].str.contains("WES")]
anz_t5 = filtered_stocks[filtered_stocks['File'].str.contains("ANZ")]
bhp_t5 = filtered_stocks[filtered_stocks['File'].str.contains("BHP")]
cba_t5 = filtered_stocks[filtered_stocks['File'].str.contains("CBA")]
rio_t5 = filtered_stocks[filtered_stocks['File'].str.contains("RIO")]
```

In [60]:
```
# For WES
wes_t5[['MAV5','MAV20']].tail(5)
```

Out[60]:

	MAV5	MAV20
4224	49.764	49.4980
4225	49.864	49.5005
4226	49.972	49.5400
4227	50.022	49.5730
4228	50.062	49.6090

In [61]:
```
# For ANZ
anz_t5[['MAV5','MAV20']].tail(5)
```

Out[61]:

	MAV5	MAV20
4231	29.156	29.1310
4232	29.074	29.1175
4233	29.054	29.1450
4234	29.002	29.1555
4235	29.050	29.1595

In [62]:
```
# For BHP
bhp_t5[['MAV5','MAV20']].tail(5)
```

Out[62]:

	MAV5	MAV20
4235	34.450	33.7615
4236	34.232	33.7335
4237	34.062	33.7350
4238	33.968	33.7730
4239	33.992	33.7970

In [63]:
```
# For CBA
cba_t5[['MAV5','MAV20']].tail(5)
```

Out[63]:

	MAV5	MAV20
4229	74.122	74.9180
4230	73.802	74.7830
4231	73.762	74.7795
4232	73.998	74.8135
4233	74.342	74.8155

In [64]:
```
# For RIO
rio_t5[['MAV5','MAV20']].tail(5)
```

Out[64]:

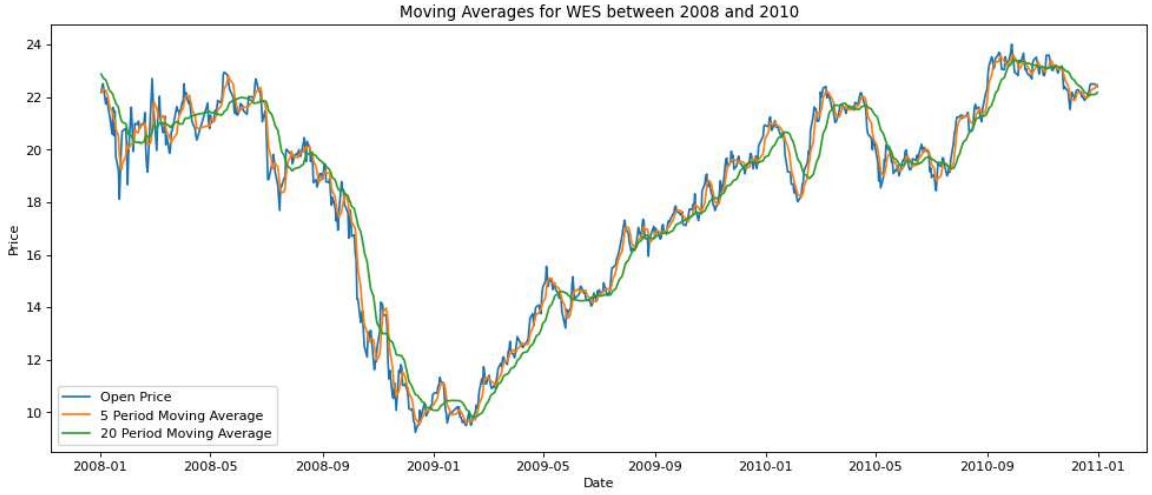
	MAV5	MAV20
4234	77.04448	77.705415
4235	76.19038	77.447830
4236	75.29754	77.250765
4237	74.87042	77.162865
4238	75.02888	77.074650

Graphing each stock by the opening price and moving averages

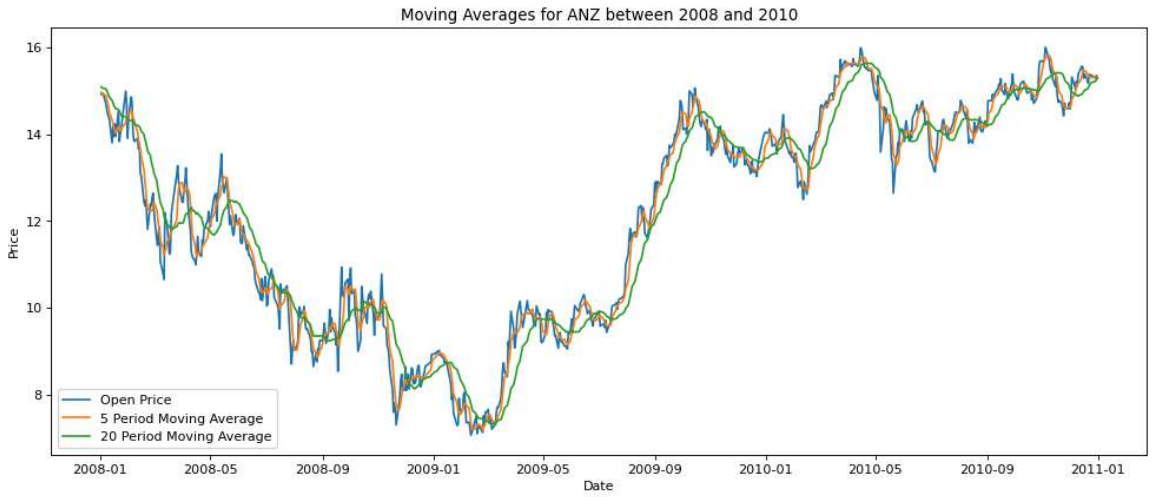
In [65]:
```
# Placing data into a new dataframe that accomodates the date 2008 to 2010
filtered_stocks_date = filtered_stocks

filtered_stocks_date.Date = pd.to_datetime(filtered_stocks_date.Date)
filtered_stocks_date.set_index('Date', inplace=True, drop=True)
filtered_stocks_date.index = pd.to_datetime(filtered_stocks_date.index)

filtered_stocks_date = filtered_stocks_date.loc['2008-01-01': '2010-12-31']

# Code utilized from: https://www.youtube.com/watch?v=_LWjaAiKaf8
```

In [66]:
```
# Selecting each stock seperately
wes = filtered_stocks_date[filtered_stocks_date['File'].str.contains("WE
S")]
anz = filtered_stocks_date[filtered_stocks_date['File'].str.contains("AN
Z")]
bhp = filtered_stocks_date[filtered_stocks_date['File'].str.contains("BH
P")]
cba = filtered_stocks_date[filtered_stocks_date['File'].str.contains("CB
A")]
rio = filtered_stocks_date[filtered_stocks_date['File'].str.contains("RI
O")]
```

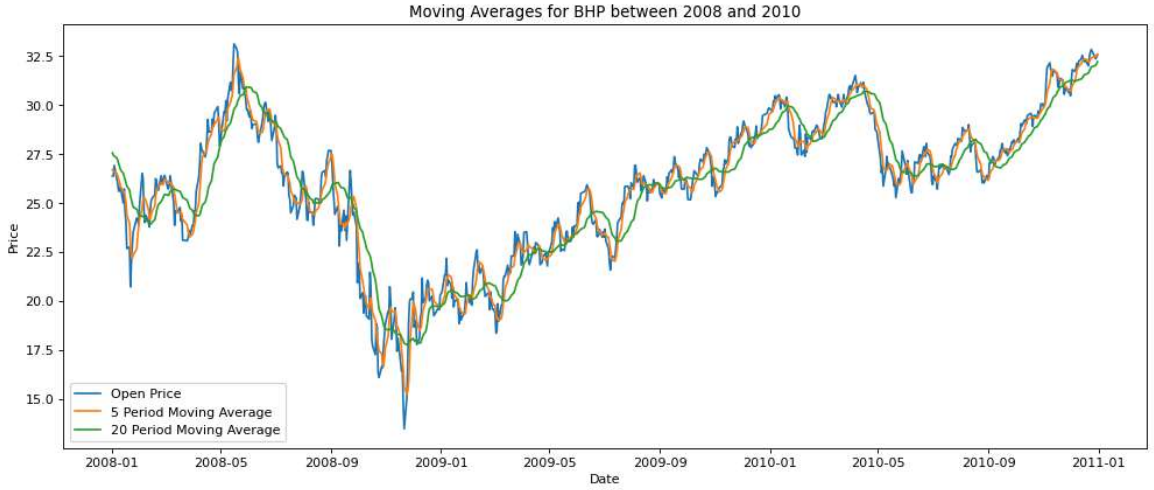
In [67]:
```
# WES opening price and moving averages graph
figure(num=None, figsize=(15, 6), dpi=80, facecolor='w', edgecolor='k')
plt.plot(wes['Open'], label='Open Price')
plt.plot(wes['MAV5'], label='5 Period Moving Average')
plt.plot(wes['MAV20'], label='20 Period Moving Average')
plt.title('Moving Averages for WES between 2008 and 2010')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```



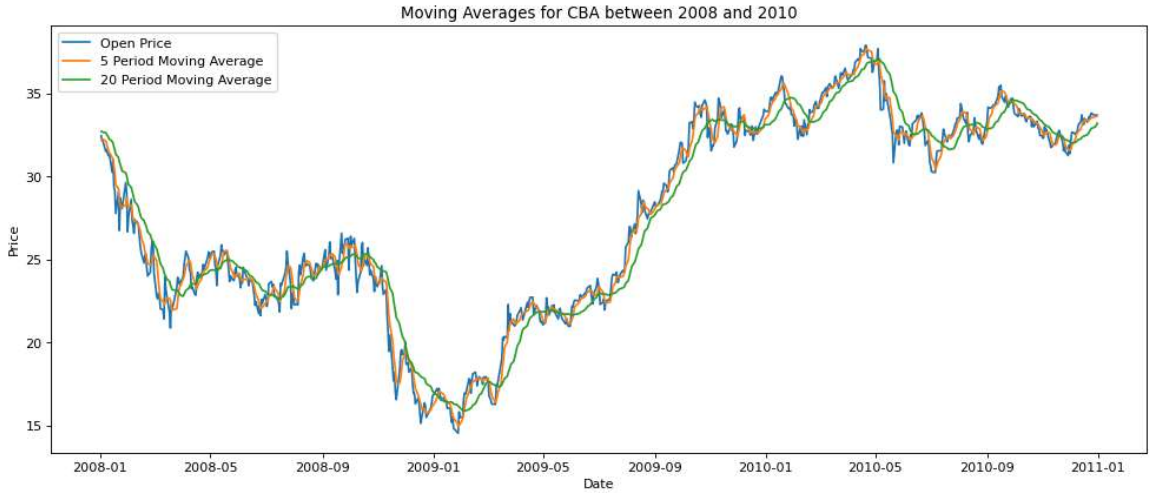
In [68]:
```
# ANZ opening price and moving averages graph
figure(num=None, figsize=(15, 6), dpi=80, facecolor='w', edgecolor='k')
plt.plot(anz['Open'], label='Open Price')
plt.plot(anz['MAV5'], label='5 Period Moving Average')
plt.plot(anz['MAV20'], label='20 Period Moving Average')
plt.title('Moving Averages for ANZ between 2008 and 2010')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```



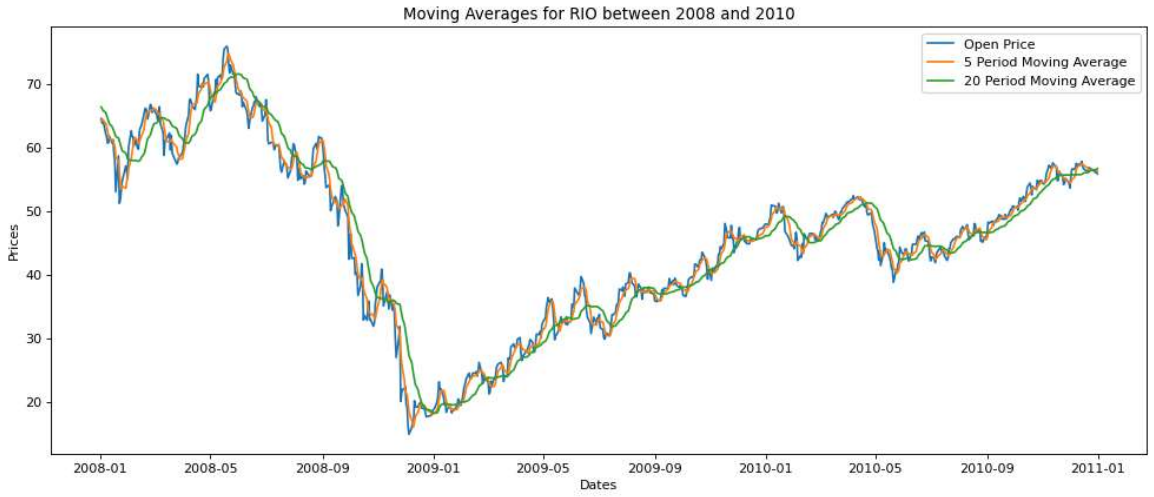
In [69]:
```
# BHP opening price and moving averages graph
figure(num=None, figsize=(15, 6), dpi=80, facecolor='w', edgecolor='k')
plt.plot(bhp['Open'], label='Open Price')
plt.plot(bhp['MAV5'], label='5 Period Moving Average')
plt.plot(bhp['MAV20'], label='20 Period Moving Average')
plt.title('Moving Averages for BHP between 2008 and 2010')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```



In [70]:
```
# CBA opening price and moving averages graph
figure(num=None, figsize=(15, 6), dpi=80, facecolor='w', edgecolor='k')
plt.plot(cba['Open'], label='Open Price')
plt.plot(cba['MAV5'], label='5 Period Moving Average')
plt.plot(cba['MAV20'], label='20 Period Moving Average')
plt.title('Moving Averages for CBA between 2008 and 2010')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```



In [71]:
```
# RIO opening price and moving averages graph
figure(num=None, figsize=(15, 6), dpi=80, facecolor='w', edgecolor='k')
plt.plot(rio['Open'], label='Open Price')
plt.plot(rio['MAV5'], label='5 Period Moving Average')
plt.plot(rio['MAV20'], label='20 Period Moving Average')
plt.title('Moving Averages for RIO between 2008 and 2010')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.legend()
plt.show()
```



Strategy Implementation

In [72]:
```
# Adding all the stocks in one dataframe
wes_strat = filtered_stocks[filtered_stocks['File'].str.contains("WES")]
anz_strat = filtered_stocks[filtered_stocks['File'].str.contains("ANZ")]
bhp_strat = filtered_stocks[filtered_stocks['File'].str.contains("BHP")]
cba_strat = filtered_stocks[filtered_stocks['File'].str.contains("CBA")]
rio_strat = filtered_stocks[filtered_stocks['File'].str.contains("RIO")]

stocks_strat = pd.concat([wes_strat, anz_strat, bhp_strat, cba_strat, rio_s
trat])

# Code utilized from: https://stackoverflow.com/questions/39301465/appendin
g-one-data-frame-into-another
```

In [73]:
```
# Selecting only the opening price, closing price, MAV5 and MAV20
stocks_strat = stocks_strat.drop(['High', 'Low', 'Volume', 'Member', 'Unnam
ed: 0'], axis = 1)

# Code utilized from: https://www.geeksforgeeks.org/python-delete-rows-colu
mns-from-dataframe-using-pandas-drop/
```

In [74]:
```
# Recording buy and sell transaction of the strategy
def strategy(stocks_strat):
    buy_price = []
    sell_price = []
    flag = -1

    for i in range(len(stocks_strat)):
        if stocks_strat['MAV5'][i] < stocks_strat['MAV20'][i]:
            if flag != 1:
                buy_price.append(stocks_strat['Open'][i])
                sell_price.append(np.nan)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
        elif  stocks_strat['MAV5'][i] > stocks_strat['MAV20'][i]:
         if flag != 0:
            buy_price.append(np.nan)
            sell_price.append(stocks_strat['Open'][i])
        else:
            buy_price.append(np.nan)
            sell_price.append(np.nan)
    else:
            buy_price.append(np.nan)
            sell_price.append(np.nan)

    return (buy_price, sell_price)

# Code utilized fromL https://www.youtube.com/watch?v=SEQbb8w7VTw
```

Task 3

In [75]:
```
# Storing function data into variable
results = strategy(stocks_strat)
buy_price = results[0]
sell_price = results[1]

del buy_price[-1]
del sell_price[-1]

stocks_strat['Buy_Price'] = buy_price
stocks_strat['Sell_Price'] = sell_price
```

In [76]:
```
# Adding final columns to the dataframe stocks_strat
stocks_strat.rename(columns={'File': 'Stock'}, inplace=True)

stocks_strat['Quantity'] = 1000
stocks_strat.loc[:19, "Quantity"] = np.nan

stocks_strat['Profit'] = stocks_strat['Sell_Price'].sum() - stocks_strat['B
uy_Price'].sum()
stocks_strat.loc[:19, "Profit"] = np.nan
print(stocks_strat)
```

```
           Stock     Open    Close      MAV5      MAV20  Buy_Price  \
Date
2002-01-31   WES  10.9880  11.1336       NaN        NaN        NaN
2002-02-01   WES  11.1517  11.0786       NaN        NaN        NaN
2002-02-04   WES  11.1554  11.1325       NaN        NaN        NaN
2002-02-05   WES  11.1041  11.0972       NaN        NaN        NaN
2002-02-06   WES  11.0972  10.9407  11.09928        NaN        NaN
...          ...      ...      ...       ...        ...        ...
2018-08-06   RIO  75.1465  74.5848  77.04448  77.705415    75.1465
2018-08-07   RIO  74.2556  73.9844  76.19038  77.447830    74.2556
2018-08-08   RIO  74.6623  74.7494  75.29754  77.250765    74.6623
2018-08-09   RIO  75.1800  75.9400  74.87042  77.162865    75.1800
2018-08-10   RIO  75.9000  75.6600  75.02888  77.074650    75.9000

            Sell_Price  Quantity      Profit
Date
2002-01-31         NaN       NaN         NaN
2002-02-01         NaN       NaN         NaN
2002-02-04         NaN       NaN         NaN
2002-02-05         NaN       NaN         NaN
2002-02-06         NaN       NaN         NaN
...                ...       ...         ...
2018-08-06         NaN    1000.0  87541.7446
2018-08-07         NaN    1000.0  87541.7446
2018-08-08         NaN    1000.0  87541.7446
2018-08-09         NaN    1000.0  87541.7446
2018-08-10         NaN    1000.0  87541.7446

[21178 rows x 9 columns]
```

Overview of stocks_strat

In [77]:
```
# Stocks_strat describe
stocks_described = stocks_strat.describe().loc[['mean','50%','std','max','min']]
print(stocks_described)
```

```
            Open      Close       MAV5      MAV20  Buy_Price  Sell_Price  \
mean   28.017879  28.009648  28.015015  28.003588  27.511335   28.433033
50%    23.930700  23.932250  23.925950  23.961470  23.534400   24.354350
std    17.084839  17.082790  17.066134  17.004874  16.878374   17.232445
max    84.152400  84.007200  82.699860  81.582780  82.825700   84.152400
min     4.922500   4.900100   4.988300   5.113300   4.922500    5.173700

      Quantity        Profit
mean    1000.0  8.754174e+04
50%     1000.0  8.754174e+04
std        0.0  4.039707e-08
max     1000.0  8.754174e+04
min     1000.0  8.754174e+04
```

Calculating profit of each stock

In [78]:
```
# WES trading profit
wes_profit = stocks_strat[stocks_strat['Stock'].str.contains("WES")]
wes_profit['Profit'] = wes_profit['Sell_Price'].sum() - wes_profit['Buy_Price'].sum()
print('Total profit from WES: ', wes_profit['Profit'].iloc[-1])

# ANZ trading profit
anz_profit = stocks_strat[stocks_strat['Stock'].str.contains("ANZ")]
anz_profit['Profit'] = anz_profit['Sell_Price'].sum() - anz_profit['Buy_Price'].sum()
print('Total profit from ANZ: ', anz_profit['Profit'].iloc[-1])

# BHP trading profit
bhp_profit = stocks_strat[stocks_strat['Stock'].str.contains("BHP")]
bhp_profit['Profit'] = bhp_profit['Sell_Price'].sum() - bhp_profit['Buy_Price'].sum()
print('Total profit from BHP: ', bhp_profit['Profit'].iloc[-1])

# CBA trading profit
cba_profit = stocks_strat[stocks_strat['Stock'].str.contains("WES")]
cba_profit['Profit'] = cba_profit['Sell_Price'].sum() - cba_profit['Buy_Price'].sum()
print('Total profit from CBA: ', cba_profit['Profit'].iloc[-1])

# RIO trading profit
rio_profit = stocks_strat[stocks_strat['Stock'].str.contains("WES")]
rio_profit['Profit'] = rio_profit['Sell_Price'].sum() - rio_profit['Buy_Price'].sum()
print('Total profit from RIO: ', rio_profit['Profit'].iloc[-1])
```

```
Total profit from WES:  19213.982799999998
Total profit from ANZ:  11377.8253
Total profit from BHP:  8001.5336000000025
Total profit from CBA:  19213.982799999998
Total profit from RIO:  19213.982799999998
```

In [79]:
```
# Number of trades
buy_trades = stocks_strat['Buy_Price'].shape[0] - stocks_strat['Buy_Pric
e'].isnull().sum()

sell_trades = stocks_strat['Sell_Price'].shape[0] - stocks_strat['Sell_Pric
e'].isnull().sum()

total_trades = sell_trades - buy_trades

print('Total number of trades: ', total_trades)
```

```
Total number of trades:  2781
```

Comments on the strategy

The strategy involves purchasing stock whenever the 5 moving average of a stock crosses over the 20 moving average and selling when it crosses back over would generate a profit of 87541 dollars based on the historical prices of the stocks, WES, ANZ, CBA, BHP and RIO.

It is evident that the 3 stocks (WES, CBA and RIO) returned the same amount of profit of 19213.98 dollars, while the second best performing stock was ANZ with a return of 11377.825 dollars. The worst performing stock was BHP which generated a return of 8001.53 dollars.

This strategy could be used in the future if the trader studies the market and invests into the highest performing stocks as the strategy did generate a profit.

However, the strategy could be improved by following Yahoo Finance's suggested strategy of having a crossover of 13 days and a 48.5 days. This strategy utilizes the 'Golden Cross' where returns are at a 4.9 percent gain.

Reference: https://finance.yahoo.com/news/study-determines-best-moving-average-195042216.html (https://finance.yahoo.com/news/study-determines-best-moving-average-195042216.html)