Task - 1:

Purpose

We want to process the data feed of stock A and stock B's price to enable us to analyse when trading for the stock should occur.

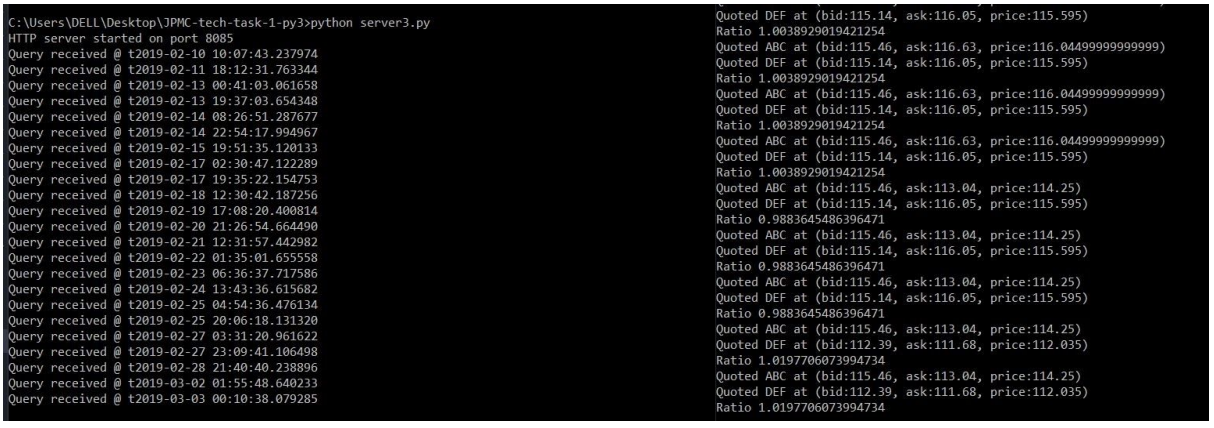
Acceptance Criteria

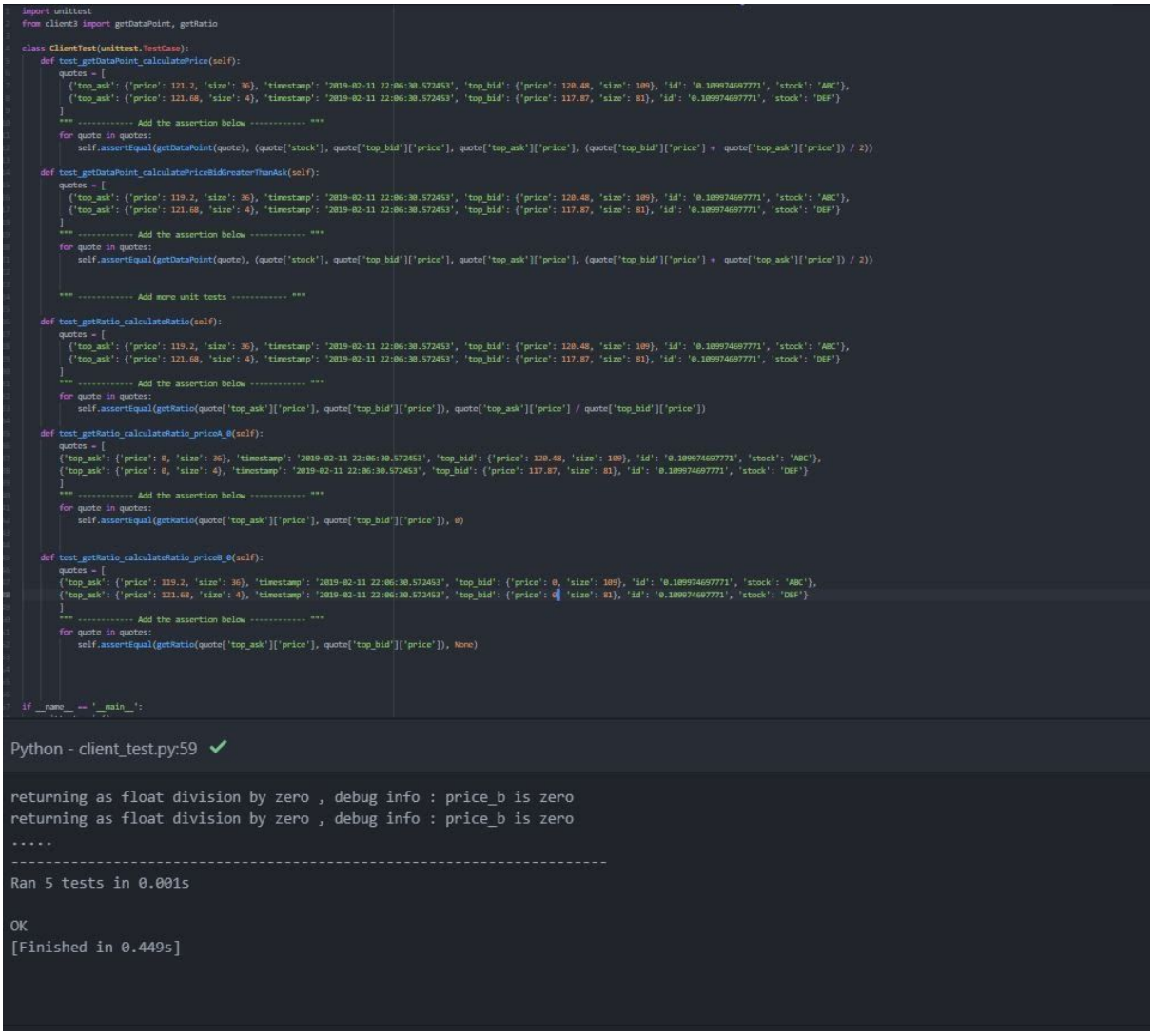
- *getDataPoint* function should return the correct tuple of stock name, bid_price, ask_price and price. Note: price of a stock = average of bid and ask
- *getRatio* function should return the ratio of the two stock prices
- main function should output correct stock info, prices and ratio
- Upload a git patch file as the submission to this task
- Bonus: All unit tests inside client_test.py, added/existing have to pass

My achievements :

- I made the necessary changes in the functions named *getDataPoint* and *getRatio* to get the data pipeline ready.
- I wrote / added several test cases for testing the functions. This really helped.

Task - 2 :





```
import unittest
from client3 import getDataPoint, getRatio

class ClientTest(unittest.TestCase):
    def test_getDataPoint_calculatePrice(self):
        quotes = [
          {'top_ask': {'price': 121.2, 'size': 36}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 120.48, 'size': 109}, 'id': '0.109974697771', 'stock': 'ABC'},
          {'top_ask': {'price': 121.68, 'size': 4}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 117.87, 'size': 81}, 'id': '0.109974697771', 'stock': 'DEF'}
        ]
        """ ----------- Add the assertion below ----------- """
        for quote in quotes:
            self.assertEqual(getDataPoint(quote), (quote['stock'], quote['top_bid']['price'], quote['top_ask']['price'], (quote['top_bid']['price'] +  quote['top_ask']['price']) / 2))

    def test_getDataPoint_calculatePriceBidGreaterThanAsk(self):
        quotes = [
          {'top_ask': {'price': 119.2, 'size': 36}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 120.48, 'size': 109}, 'id': '0.109974697771', 'stock': 'ABC'},
          {'top_ask': {'price': 121.68, 'size': 4}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 117.87, 'size': 81}, 'id': '0.109974697771', 'stock': 'DEF'}
        ]
        """ ----------- Add the assertion below ----------- """
        for quote in quotes:
            self.assertEqual(getDataPoint(quote), (quote['stock'], quote['top_bid']['price'], quote['top_ask']['price'], (quote['top_bid']['price'] +  quote['top_ask']['price']) / 2))


        """ ----------- Add more unit tests ----------- """

    def test_getRatio_calculateRatio(self):
        quotes = [
          {'top_ask': {'price': 119.2, 'size': 36}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 120.48, 'size': 109}, 'id': '0.109974697771', 'stock': 'ABC'},
          {'top_ask': {'price': 121.68, 'size': 4}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 117.87, 'size': 81}, 'id': '0.109974697771', 'stock': 'DEF'}
        ]
        """ ----------- Add the assertion below ----------- """
        for quote in quotes:
            self.assertEqual(getRatio(quote['top_ask']['price'], quote['top_bid']['price']), quote['top_ask']['price'] / quote['top_bid']['price'])

    def test_getRatio_calculateRatio_priceA_0(self):
        quotes = [
        {'top_ask': {'price': 0, 'size': 36}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 120.48, 'size': 109}, 'id': '0.109974697771', 'stock': 'ABC'},
        {'top_ask': {'price': 0, 'size': 4}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 117.87, 'size': 81}, 'id': '0.109974697771', 'stock': 'DEF'}
        ]
        """ ----------- Add the assertion below ----------- """
        for quote in quotes:
            self.assertEqual(getRatio(quote['top_ask']['price'], quote['top_bid']['price']), 0)


    def test_getRatio_calculateRatio_priceB_0(self):
        quotes = [
        {'top_ask': {'price': 119.2, 'size': 36}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 0, 'size': 109}, 'id': '0.109974697771', 'stock': 'ABC'},
        {'top_ask': {'price': 121.68, 'size': 4}, 'timestamp': '2019-02-11 22:06:30.572453', 'top_bid': {'price': 0, 'size': 81}, 'id': '0.109974697771', 'stock': 'DEF'}
        ]
        """ ----------- Add the assertion below ----------- """
        for quote in quotes:
            self.assertEqual(getRatio(quote['top_ask']['price'], quote['top_bid']['price']), None)



if __name__ == '__main__':
```

Python - client_test.py:59 ✔

```
returning as float division by zero , debug info : price_b is zero
returning as float division by zero , debug info : price_b is zero
.....
----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK
[Finished in 0.449s]
```

Task - 2:

Purpose:

The objective of this task will be for you to fix the client-side web application so that it displays a graph that automatically updates as it gets data from the server application (*see Before and After images below*) Currently, the web application only gets data every time you click on the 'Start Streaming Data' button and does not aggregate duplicated data.

Acceptance Criteria:

- This ticket is done when the graph displayed in the client-side web application is a continuously updating line graph whose y axis is the stock’s top_ask_price and the x-axis

is the timestamp of the stock. The continuous updates to the graph should be the result of continuous requests and responses to and from the server for the stock data.

- This ticket is done when the graph is also able to aggregate duplicated data retrieved from the server.

My achievements :

- Added properly comments to the code integrated into the react app.

```
 * Get new data from server and update the state with the new data
 */
getDataFromServer() {
  let x = 0;
  const interval = setInterval(() => {
    DataStreamer.getData((serverResponds: ServerRespond[]) => {
      // Update the state by creating a new array of data that consists of
      // Previous data in the state and the new data from server
      this.setState( {
          data: serverResponds,
          showGraph: true,
      });
  });
  x++;
  if (x > 1000) {
    clearInterval(interval);
  }
}, 100);

/**
 * Render the App react component
 */
render() {
```

```
}
if (this.table) {
  // Load the `table` in the `<perspective-viewer>` DOM reference.

  // Add more Perspective configurations here.
  elem.load(this.table);
  elem.setAttribute('view', 'y_line'); // this is the view of data
  elem.setAttribute('column-pivots', '["stock"]'); // this allows use to distinguish stocks
  elem.setAttribute('row-pivots', '["timestamp"]'); // this maps each datapoint based on timestamp
  elem.setAttribute('columns', '["top_ask_price"]'); // it allows to only focus on a particular part of a stock's data
  // aggregates helps us to handle the duplicated data
  elem.setAttribute('aggregates',`
    {"stock" : "distinct_count",
      "top_ask_price":"avg",
      "top_bid_price":"avg",
      "timestamp":"distinct_count"}`);
}
```

Task - 3:

Purpose

You will use perspective to generate a live graph that displays the data feed in a clear and visually appealing way for traders to monitor this trading strategy.

Recall that the purpose of this graph is to monitor and determine when a trading opportunity may arise as a result of the temporary weakening of a correlation between two stock prices. Given this graph, the trader should be able to quickly and easily notice when the ratio moves too far from the average historical correlation. In the first instance, we'll assume that threshold is +/-10% of the 12 month historical average ratio.

Acceptance Criteria

- This ticket is done when the numbers from the python script render properly in the live perspective graph. This means the ratio between the two stock prices is tracked and displayed. The upper and lower bounds must be shown on the graph too. And finally, alerts are shown whenever these bounds are crossed by the ratio (the guide below will also give more detail and visuals to help you understand these requirements better)

- This ticket is done when a patch file is uploaded along with a video or audio explanation of the final chart you have produced

My achievements :

- Added properly comments to the code integrated into the react app.

```
componentDidMount() {
  // Get element from the DOM.
  const elem = document.getElementsByTagName('perspective-viewer')[0] as unknown

  const schema = {
    price_abc: 'float', // needed for calcuting the ratios
    price_def: 'float',  // needed for calculating the ratios
    ratio: 'float',   // this will show the ratio between stocks
    timestamp: 'date',  // calcuation is w.r.t time
    upper_bound: 'float',  // the upper threshold
    lower_bound: 'float',  // the lower threshold
    trigger_alert: 'float',  // the alert when the correlations starts become we
  };

  if (window.perspective && window.perspective.worker()) {
    this.table = window.perspective.worker().table(schema);
  }
  if (this.table) {
    // Load the `table` in the `<perspective-viewer>` DOM reference.
    // now we don't need column-pivots , as now we are tracking ratios
    elem.load(this.table);
    elem.setAttribute('view', 'y_line'); // this is the view
    elem.setAttribute('row-pivots', '["timestamp"]');
    elem.setAttribute('columns', '["ratio", "lower_bound", "upper_bound", "trigg
    elem.setAttribute('aggregates', JSON.stringify({
      price_abc: 'float',
```

```
export class DataManipulator {
  static generateRow(serverResponds: ServerRespond[]): Row {
    const priceABC = (serverResponds[0].top_ask.price + serverResponds[0].top_bid.pri
    const priceDEF = (serverResponds[1].top_ask.price + serverResponds[1].top_bid.pri
    const ratio = priceABC / priceDEF;
    const upperBound = 1 + 0.05;
    const lowerBound = 1 - 0.05;
      return {
        price_abc: priceABC,
        price_def: priceDEF,
        ratio,
        timestamp: serverResponds[0].timestamp > serverResponds[1].timestamp ?
          serverResponds[0].timestamp : serverResponds[1].timestamp,
        upper_bound: upperBound, // this is dynamic
        lower_bound: lowerBound, // this is dynamic
        trigger_alert: (ratio > upperBound || ratio < lowerBound) ? ratio : undefined
      };
    }
  }
```