Entelligent SQL Assessment

Chris Lundberg

Assumptions

- The returns reported in the field `q_ending_total_return` are returns from the previous quarter end to the date corresponding to the observation. For example, if `t_date` is 6/30/2020 and `q_end_total_return` is 13%, the stock in question returned 13% between 3/31/2020 and 6/30/2020.
- The the e_scores in `score` are scores as of the corresponding `t_date`.
- The weights in `weight` are weights as of the corresponding `t_date`.
- The 1% risk free rate is an annual rate, and therefore I used a risk free rate of 0.25% in the sharpe ratio calculations so that the risk free rate and the returns are both stated on a quarterly basis.

Query I

This query will return the continuously compounded return for each stock between 12/31/2018 and 12/31/2019.

```
SELECT ticker,
       total_return
FROM   (SELECT t_date,
               ticker,
               exp(sum(ln(1 + q_ending_total_return))
                     OVER (
                       partition BY ticker
                       ORDER BY t_date)) - 1 total_return
        FROM   performance
        WHERE  t_date > '2018-12-31'
               AND t_date <= '2019-12-31') AS returns
WHERE  t_date = '2019-12-31';
```

Query II

This query will return the one year rolling average and one year rolling standard deviation of e_scores, where both statistics are computed across all stocks.

```
SELECT   t_date date,
         (
                SELECT avg(score)
                FROM   e_scores AS temp
                WHERE  temp.t_date <= e_scores.t_date
                AND    temp.t_date >= e_scores.t_date - INTERVAL '1 year') avg_score,
         (
                SELECT stddev_samp(score)
                FROM   e_scores AS temp
                WHERE  temp.t_date <= e_scores.t_date
                AND    temp.t_date >= e_scores.t_date - INTERVAL '1 year') stddev_score
FROM     e_scores
GROUP BY date
ORDER BY date;
```

Query III

This query will return the weighted average return and weighted average e_score for each benchmark for each quarter.

```
SELECT constituents.t_date                date,
       benchmark,
       sum(weight * q_ending_total_return) bench_return,
       sum(weight * score)                 bench_e_score
FROM   constituents
       LEFT JOIN performance
              ON ( constituents.t_date = performance.t_date
                   AND constituents.ticker = performance.ticker )
       LEFT JOIN e_scores
              ON ( constituents.t_date = e_scores.t_date
                   AND constituents.ticker = e_scores.ticker )
GROUP  BY date,
          benchmark
ORDER  BY date;
```

Query IV

This query will return the quarterly performance of an equal weight portfolio of stocks in the S&P with below average e_scores in the previous quarter.

```
SELECT   date,
         avg(return) equal_weight_return
FROM     (
                  SELECT    constituents.t_date date,
                            performance.ticker,
                            q_ending_total_return RETURN
                  FROM      (
                                     SELECT   t_date + INTERVAL '3 months' date,
                                              avg(score)                   avg_score
                                     FROM     e_scores
                                     GROUP BY t_date) AS scores
                  LEFT JOIN constituents
                  ON        date_part('month', scores.date) = date_part('month', constituents.t_date)
                  AND       date_part('year', scores.date) = date_part('year', constituents.t_date)
                  LEFT JOIN e_scores
                  ON        scores.date = e_scores.t_date + INTERVAL '3 months'
                  AND       constituents.ticker = e_scores.ticker
                  LEFT JOIN performance
                  ON        constituents.t_date = performance.t_date
                  AND       constituents.ticker = performance.ticker
                  WHERE     benchmark = 'SPY-US'
                  AND       score < avg_score ) AS returns
GROUP BY date;
```

Query V

This query will return the quarterly performance of an equal weight portfolio of stocks in the S&P with below average e_scores over the previous four quarters.

```
SELECT   date,
         avg(q_ending_total_return) equal_weight_return
FROM     (
                  SELECT    performance.t_date date,
                            performance.ticker,
                            q_ending_total_return
                  FROM      (
                                      SELECT   t_date + INTERVAL '3 months' date,
                                               (SELECT avg(score)
                                                FROM   e_scores AS temp
                                                WHERE  temp.t_date <= e_scores.t_date
                                                AND    temp.t_date >= e_scores.t_date - INTERVAL '1 year')
                                                avg_score
                                      FROM     e_scores
                                      GROUP BY t_date) AS scores
                  LEFT JOIN performance
                  ON        date_part('month', scores.date) = date_part('month', performance.t_date)
                  AND       date_part('year', scores.date) = date_part('year', performance.t_date)
                  LEFT JOIN
                            (
                                      SELECT   t_date + INTERVAL '3 months' date,
                                               ticker,
                                               avg(score) over
                                               (partition BY ticker
                                               ORDER BY t_date rows
                                               BETWEEN 3 preceding AND current row) avg_score_four_qtr
                                      FROM     e_scores) AS stocks
                  ON        date_part('month', performance.t_date) = date_part('month', stocks.date)
                  AND       date_part('year', performance.t_date) = date_part('year', stocks.date)
                  AND       performance.ticker = stocks.ticker
                  LEFT JOIN constituents
                  ON        performance.t_date = constituents.t_date
                  AND       performance.ticker = constituents.ticker
                  WHERE     benchmark = 'SPY-US'
                  AND       avg_score_four_qtr < avg_score) AS returns
GROUP BY date;
```

Query VI

This query will return the sharpe ratio for each stock (the `sharpe_ratio` field) and the sharpe ratio for the S&P 500 (the `sp500_sharpe_ratio` field).

```
SELECT ticker,
       sharpe_ratio,
       sp500_sharpe_ratio
FROM   (SELECT ( avg(sp_return) - 0.0025 ) / stddev_samp(sp_return)
               sp500_sharpe_ratio
        FROM   (SELECT constituents.t_date                 date,
                       benchmark,
                       sum(weight * q_ending_total_return) sp_return
                FROM   constituents
                       LEFT JOIN performance
                              ON ( constituents.t_date = performance.t_date
                                   AND constituents.ticker = performance.ticker
                                 )
                WHERE  benchmark = 'SPY-US'
                GROUP  BY date,
                          benchmark
                ORDER  BY date) AS sp) AS bench
       CROSS JOIN (SELECT ticker,
                          ( avg(q_ending_total_return) - 0.0025 ) /
                          stddev_samp(q_ending_total_return)
                          sharpe_ratio
                   FROM   performance
                   GROUP  BY ticker) AS stocks
WHERE  sharpe_ratio IS NOT NULL
ORDER  BY sharpe_ratio DESC;
```