ACE2.0 Data Processing Module

Wut Hmone Hnin Hlaing (wut.hmone@tmrnd.com.my)

October 15, 2018

ace2/advisory/___init___.py

```
__all__ = ['threshold','smartqueue']

import ace2.advisory.threshold
import ace2.advisory.smartqueue
```

ace2/advisory/smartqueue_ast.py

```
import ace2
import pandas as pd
import json
import requests as rq


def getqmsthresholdata(companyname, url, port):
    request = rq.get('http://'+url+':'+port+'/ace/api/advisory/threshold?company='+companyname,
                     headers={'Content-type': 'application/json'})
    value = request.text
    data = json.loads(value)
    if data['error'] == False:
        data = data['Threshold']
        data_stored = []
        if len(data) == 0:
            return data_stored
        else:
            data = data[0]['data']
            company_data = data['components']
            if len(company_data) > 0:
                for each in company_data:
                    if each['component'] == 'qms':
                        qms_data = dict()
                        qms_data['component'] = 'qms'
                        oneweek = each['oneweek']
                        oneweek_dict = dict()
                        oneweek_tmp = pd.io.json.json_normalize(oneweek['tmp'])
                        oneweek_sta = pd.io.json.json_normalize(oneweek['sta'])
```

```
29                          oneweek_mal = pd.io.json.json_normalize(oneweek['
                                mal'])
30                          oneweek_dict['tmp'] = oneweek_tmp
31                          oneweek_dict['sta'] = oneweek_sta
32                          oneweek_dict['mal'] = oneweek_mal
33                          qms_data['oneweek'] = oneweek_dict
34
35                          onemonth = each['onemonth']
36                          onemonth_dict = dict()
37                          onemonth_tmp = pd.io.json.json_normalize(onemonth[
                                'tmp'])
38                          onemonth_sta = pd.io.json.json_normalize(onemonth[
                                'sta'])
39                          onemonth_mal = pd.io.json.json_normalize(onemonth[
                                'mal'])
40                          onemonth_dict['tmp'] = onemonth_tmp
41                          onemonth_dict['sta'] = onemonth_sta
42                          onemonth_dict['mal'] = onemonth_mal
43                          qms_data['onemonth'] = onemonth_dict
44
45                          threemonths = each['threemonths']
46                          threemonths_dict = dict()
47                          threemonths_tmp = pd.io.json.json_normalize(
                                threemonths['tmp'])
48                          threemonths_sta = pd.io.json.json_normalize(
                                threemonths['sta'])
49                          threemonths_mal = pd.io.json.json_normalize(
                                threemonths['mal'])
50                          threemonths_dict['tmp'] = threemonths_tmp
51                          threemonths_dict['sta'] = threemonths_sta
52                          threemonths_dict['mal'] = threemonths_mal
53                          qms_data['threemonths'] = threemonths_dict
54                          data_stored.append(qms_data)
55                      if each['component'] == 'wifi':
56                          wifi_data = dict()
57                          wifi_data['component'] = 'wifi'
58                          data_stored.append(wifi_data)
59                      if each['component'] == 'video':
60                          video_data = dict()
61                          video_data['component'] = 'video'
62                          data_stored.append(video_data)
63                      if each['component'] == 'engage':
64                          engage_data = dict()
65                          engage_data['component'] = 'engage'
66                          data_stored.append(engage_data)
67                      if each['component'] == 'target':
68                          target_data = dict()
69                          target_data['component'] = 'target'
70                          data_stored.append(target_data)
71              return data_stored
72
73
74  def getqmsdatafromday(date, hour, min_, url, port):
75      request = rq.get('http://'+url+':'+port+'/ace/api/v1/qms?date=' +
```

```
                        date, headers={'Content-type': 'application/json'})
    value = request.text
    qmsdata = json.loads(value)

    if qmsdata['error'] == False:
        data = qmsdata['qms']

        if len(data) == 0:
            return pd.DataFrame()
        else:
            timestamp = date + " " + hour + ":" + min_
            # print(timestamp)
            date_timestamp = ace2.datetime.strptime(timestamp, "%Y%m%d %H
                :%M")
            str_timestamp = date_timestamp.strftime('%s')
            qms_avg = pd.DataFrame()
            qms_rank = pd.DataFrame()
            for each in data:
                premise = each['premise']
                premise_qms = each['qmsdata']
                premise_qms = pd.io.json.json_normalize(premise_qms)
                premise_qms['premisename'] = premise
                premise_qms['time_stamp'] = premise_qms['time_stamp'].
                    apply(
                        lambda x: ace2.normalizeTimeStamp(x)).astype(int)
                premise_qms['serveAvg'] = premise_qms['serveAvg'].apply(
                        lambda x: ace2.convertTime(x))
                premise_qms['waitingAvg'] = premise_qms['waitingAvg'].
                    apply(
                        lambda x: ace2.convertTime(x))
                premise_qms['waitingCount'] = premise_qms['waitingCount'].
                    astype(float)

                data1 = premise_qms.loc[premise_qms['time_stamp']
                                        == int(str_timestamp)-3600]  # 1
                                            hour back
                data2 = premise_qms.loc[premise_qms['time_stamp'] == int(
                    str_timestamp)]
                data = data1.append(data2, ignore_index=True)
                qms_avg = qms_avg.append(data, ignore_index=True)
                qms_rank = qms_rank.append(data2, ignore_index=True)

            qms_avg.fillna(0, inplace=True)
            qms_rank.fillna(0, inplace=True)

            return (qms_avg, qms_rank)


def postmessagedata(data, url, port, companyname):
    url = 'http://'+url+':'+port+'/ace/api/advisory?company='+companyname
    headers = {'Content-type': 'application/json'}
    r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
    print('Post :', r.text)

```

```
124
125 def main(date, hour, min_):
126     source = ace2.read_json('../input.json')
127     url = source['url']
128     port = source['port']
129     companydetails = ace2.pickpremisebycom(url, port)
130     premisedata = ace2.getpremises(url, port)
131     timestamp = date + " " + hour + ":" + min_
132     # print(timestamp)
133     date_timestamp = ace2.datetime.strptime(timestamp, "%Y%m%d %H:%M")
134     time_stamp = date_timestamp.strftime('%s')
135
136     for each in companydetails:
137         # TMpoint level
138         companyname = each['name']
139         data = getqmsthresholdata(companyname, url, port)
140         if len(data) > 0:
141             for each in data:
142                 if each['component'] == 'qms':
143                     oneweek = each['oneweek']
144                     onemonth = each['onemonth']
145                     threemonths = each['threemonths']
146                     threemonths_tmp = threemonths['tmp']
147                     onemonth_tmp = onemonth['tmp']
148                     oneweek_tmp = oneweek['tmp']
149
150                     qms_avg, qms_rank = getqmsdatafromday(date, hour, min_
                        , url, port)
151                     data_group = qms_avg.groupby(by=qms_avg.premisename,
                        as_index=False)
152
153                     # Tmpoint Level
154                     ast_message = []
155                     awt_message = []
156                     vis_message = []
157
158                     for key, value in data_group:
159                         group = pd.DataFrame(value)
160                         group = group.reset_index()
161
162                         threshold_df = threemonths_tmp.loc[threemonths_tmp
                            ['premisename'] == key]
163
164                         # TMpoint visitor
165                         if int(hour) > 15:
166                             oneweek_df = oneweek_tmp.loc[oneweek_tmp['
                                premisename'] == key]
167                             onemonth_df = onemonth_tmp.loc[onemonth_tmp['
                                premisename'] == key]
168                             threemonths_df = threemonths_tmp.loc[
                                threemonths_tmp['premisename'] == key]
169                             if len(oneweek_df) > 0 and len(onemonth_df) >
                                0 and len(threemonths_df) > 0:
```

```
                    oneweek_threshold = int(oneweek_df.iloc
                        [0]['tol_serve'])
                    minus_oneweek = oneweek_threshold - (
                        oneweek_threshold * 0.25)

                    onemonth_threshold = int(onemonth_df.iloc
                        [0]['tol_serve'])
                    minus_onemonth = onemonth_threshold - (
                        onemonth_threshold * 0.25)

                    threemonths_threshold = int(threemonths_df
                        .iloc[0]['tol_serve'])
                    minus_threemonths = threemonths_threshold
                        - \
                         (threemonths_threshold * 0.25)

                    visitor = int(group.iloc[1]['serveCount'])
                    pack_vis = dict()
                    if visitor < minus_oneweek:
                        pack_vis['premisename'] = key
                        pack_vis['message'] = 'Please organise
                            an open day to attract more
                           visitors.'
                        pack_vis['timestamp'] = time_stamp

                    if visitor < minus_onemonth:
                        pack_vis['premisename'] = key
                        pack_vis['message'] = 'Please organize
                            in outlet promo to attract more
                           visitors.'
                        pack_vis['timestamp'] = time_stamp

                    if visitor < minus_threemonths:
                        pack_vis['premisename'] = key
                        pack_vis['message'] = 'Please be
                           advised to relocate the outlet.'
                        pack_vis['timestamp'] = time_stamp

                    if len(pack_vis) > 0:
                        vis_message.append(pack_vis)

                # AVG Servinig Time
                ast_flags = {
                    'c_plus': False,
                    'c_0': False,
                    'c_minus': False
                }
                pack_ast = dict()
                if len(threshold_df) > 0 and len(group) == 2:
                    # Avg Servinig Time
                    p1_ast = group.iloc[0]['serveAvg']
                    p2_ast = group.iloc[1]['serveAvg']
                    active_counter = group.iloc[1]['activeCounter'
                        ]
```

```
212
213                         trend = 'up' if p1_ast < p2_ast else 'down'
214                         if p1_ast == p2_ast:
215                             trend = 'same'
216                         threshold_ast = threshold_df.iloc[0]['serveAvg
                                ']
217                         # if less than 5 min, make it avg to 5 min
218                         if threshold_ast < 300.0:
219                             threshold_ast = 300.0
220                         plus_threshold_ast = threshold_ast + (
                                threshold_ast * 0.25)
221                         minus_threshold_ast = threshold_ast - (
                                threshold_ast * 0.25)
222
223                         if (p2_ast >= plus_threshold_ast and
                                plus_threshold_ast >= p1_ast) or \
224                                 (p2_ast < plus_threshold_ast and
                                        plus_threshold_ast < p1_ast):
225                             ast_flags['c_plus'] = True
226
227                         if (p2_ast >= threshold_ast and threshold_ast
                                >= p1_ast) or \
228                                 (p2_ast < threshold_ast and
                                        threshold_ast < p1_ast):
229                             ast_flags['c_0'] = True
230
231                         if (p2_ast >= minus_threshold_ast and
                                minus_threshold_ast >= p1_ast)or \
232                                 (p2_ast < minus_threshold_ast and
                                        minus_threshold_ast < p1_ast):
233                             ast_flags['c_minus'] = True
234
235                         if trend == 'up':
236                             # 001
237                             if ast_flags['c_minus'] == True and
                                    ast_flags['c_0'] == False and \
238                                         ast_flags['c_plus'] == False and
                                                p2_ast > 0:
239                                 pack_ast['premisename'] = key
240                                 pack_ast['message'] = 'The average
                                        serving time keeps on increasing.
                                        Please be alert.'
241                                 pack_ast['msg_flag'] = 'yellow'
242                                 pack_ast['timestamp'] = time_stamp
243                             # 010
244                             if ast_flags['c_minus'] == False and
                                    ast_flags['c_0'] == True and \
245                                         ast_flags['c_plus'] == False and
                                                p2_ast > 0:
246                                 pack_ast['premisename'] = key
247                                 pack_ast['message'] = 'The average
                                        serving time keeps on increasing.
                                        Please be alert.'
248                                 pack_ast['msg_flag'] = 'yellow'
```

```
249                         pack_ast['timestamp'] = time_stamp
250                     # 011
251                     if ast_flags['c_minus'] == True and
                           ast_flags['c_0'] == True and \
252                             ast_flags['c_plus'] == False and
                                   p2_ast > 0:
253                         pack_ast['premisename'] = key
254                         pack_ast['message'] = 'The average
                               serving time keeps on increasing
                               and passed the normal rate. Please
                               open another counter and supervise
                               the front desk officer.'
255                         pack_ast['msg_flag'] = 'red'
256                         pack_ast['timestamp'] = time_stamp
257                     # 100
258                     if ast_flags['c_minus'] == False and
                           ast_flags['c_0'] == False and \
259                             ast_flags['c_plus'] == True and
                                   p2_ast > 0:
260                         pack_ast['premisename'] = key
261                         pack_ast['message'] = 'The average
                               serving is far above or exceeded
                               normal rate, please open all
                               counter.  Please closely monitor
                               front desk officer.'
262                         pack_ast['msg_flag'] = 'red'
263                         pack_ast['timestamp'] = time_stamp
264                     # 110
265                     if ast_flags['c_minus'] == False and
                           ast_flags['c_0'] == True and \
266                             ast_flags['c_plus'] == True and
                                   p2_ast > 0:
267                         pack_ast['premisename'] = key
268                         pack_ast['message'] = 'The average
                               serving is far above or exceeded
                               normal rate, please open all
                               counter.  Please closely monitor
                               front desk officer.'
269                         pack_ast['msg_flag'] = 'red'
270                         pack_ast['timestamp'] = time_stamp
271                     # 111
272                     if ast_flags['c_minus'] == True and
                           ast_flags['c_0'] == True and \
273                             ast_flags['c_plus'] == True and
                                   p2_ast > 0:
274                         pack_ast['premisename'] = key
275                         pack_ast['message'] = 'The average
                               serving is far above or exceeded
                               normal rate, please open all
                               counter.  Please closely monitor
                               front desk officer.'
276                         pack_ast['msg_flag'] = 'red'
277                         pack_ast['timestamp'] = time_stamp
278                     # Special Case
```

```
279                         if p2_ast > plus_threshold_ast and p1_ast
                               > plus_threshold_ast:
280                             pack_ast['premisename'] = key
281                             pack_ast['message'] = 'The average
                                   serving time keeps on increasing
                                   and passed the normal rate. Please
                                   open another counter and supervise
                                   the front desk officer.'
282                             pack_ast['msg_flag'] = 'red'
283                             pack_ast['timestamp'] = time_stamp
284                     if trend == 'down':
285                         # 001
286                         if ast_flags['c_minus'] == True and
                               ast_flags['c_0'] == False and \
287                                 ast_flags['c_plus'] == False and
                                       p2_ast > 0:
288                             pack_ast['premisename'] = key
289                             if active_counter > 3:
290                                 pack_ast['message'] = 'Great, the
                                       average serving time is normal.
                                        You can close 2 counters.'
291                             else:
292                                 pack_ast['message'] = 'Great, the
                                       average serving time is normal.
                                       '
293                             pack_ast['msg_flag'] = 'green'
294                             pack_ast['timestamp'] = time_stamp
295                         # 010
296                         if ast_flags['c_minus'] == False and
                               ast_flags['c_0'] == True and \
297                                 ast_flags['c_plus'] == False and
                                       p2_ast > 0:
298                             pack_ast['premisename'] = key
299                             pack_ast['message'] = 'Well done, the
                                   average serving time become normal'
300                             pack_ast['msg_flag'] = 'green'
301                             pack_ast['timestamp'] = time_stamp
302                         # 011
303                         if ast_flags['c_minus'] == True and
                               ast_flags['c_0'] == True and \
304                                 ast_flags['c_plus'] == False and
                                       p2_ast > 0:
305                             pack_ast['premisename'] = key
306                             pack_ast['premisename'] = key
307                             if active_counter > 3:
308                                 pack_ast['message'] = 'Great, the
                                       average serving time is normal.
                                        You can close 2 counters'
309                             else:
310                                 pack_ast['message'] = 'Great, the
                                       average serving time is normal.
                                       '
311                             pack_ast['msg_flag'] = 'green'
312                             pack_ast['timestamp'] = time_stamp
```

```
                    # 100
                    if ast_flags['c_minus'] == False and
                       ast_flags['c_0'] == False and \
                            ast_flags['c_plus'] == True and
                               p2_ast > 0:
                        pack_ast['premisename'] = key
                        pack_ast['message'] = 'Good, the
                           average serving time is decreasing,
                            keep up the good work.'
                        pack_ast['msg_flag'] = 'green'
                        pack_ast['timestamp'] = time_stamp
                    # 110
                    if ast_flags['c_minus'] == False and
                       ast_flags['c_0'] == True and \
                            ast_flags['c_plus'] == True and
                               p2_ast > 0:
                        pack_ast['premisename'] = key
                        pack_ast['message'] = 'Excellent, the
                           average serving time is decreasing
                           below the normal rate.'
                        pack_ast['msg_flag'] = 'green'
                        pack_ast['timestamp'] = time_stamp
                    # 111
                    if ast_flags['c_minus'] == True and
                       ast_flags['c_0'] == True and \
                            ast_flags['c_plus'] == True and
                               p2_ast > 0:
                        pack_ast['premisename'] = key
                        if active_counter > 3:
                            pack_ast['message'] = 'Great, the
                               average serving time is normal.
                                You can close 2 counters'
                        else:
                            pack_ast['message'] = 'Great, the
                               average serving time is normal.
                               '
                        pack_ast['msg_flag'] = 'green'
                        pack_ast['timestamp'] = time_stamp

                    # Special Case
                    if p2_ast > plus_threshold_ast and p1_ast
                       > plus_threshold_ast:
                        pack_ast['premisename'] = key
                        pack_ast['message'] = 'The average
                           serving time is decreasing, keep up
                            the good work.'
                        pack_ast['msg_flag'] = 'red'
                        pack_ast['timestamp'] = time_stamp
                # Avg Waiting Time
            if len(pack_ast) > 0:
                ast_message.append(pack_ast)

            # AVG Waiting Time
            awt_flags = {
```

```
                    'c_plus': False,
                    'c_0': False,
                    'c_minus': False
                }
                pack_awt = dict()

                if len(threshold_df) > 0 and len(group) == 2:

                    p1_awt = group.iloc[0]['waitingAvg']
                    p2_awt = group.iloc[1]['waitingAvg']
                    active_counter = group.iloc[1]['activeCounter']

                    trend = 'up' if p1_awt < p2_awt else 'down'
                    if p1_awt == p2_awt:
                        trend = 'same'
                    threshold_awt = threshold_df.iloc[0]['waitingAvg']
                    # if less than 5 min, make it avg to 5 min
                    if threshold_awt < 300.0:
                        threshold_awt = 300.0
                    plus_threshold_awt = threshold_awt + (threshold_awt * 0.25)
                    minus_threshold_awt = threshold_awt - (threshold_awt * 0.25)

                    if (p2_awt >= plus_threshold_awt and plus_threshold_awt >= p1_awt) or \
                            (p2_awt < plus_threshold_awt and plus_threshold_awt < p1_awt):
                        awt_flags['c_plus'] = True

                    if (p2_awt >= threshold_awt and threshold_awt >= p1_awt) or \
                            (p2_awt < threshold_awt and threshold_awt < p1_awt):
                        awt_flags['c_0'] = True

                    if (p2_awt >= minus_threshold_awt and minus_threshold_awt >= p1_awt)or \
                            (p2_awt < minus_threshold_awt and minus_threshold_awt < p1_awt):
                        awt_flags['c_minus'] = True

                    if trend == 'up':
                        # 001
                        if awt_flags['c_minus'] == True and awt_flags['c_0'] == False and \
                                awt_flags['c_plus'] == False and p2_awt > 0:
                            pack_awt['premisename'] = key
                            pack_awt['message'] = 'The average waiting time keeps on increasing. Please be alert.'
```

```
                pack_awt['msg_flag'] = 'yellow'
                pack_awt['timestamp'] = time_stamp
            # 010
            if awt_flags['c_minus'] == False and
               awt_flags['c_0'] == True and \
                    awt_flags['c_plus'] == False and
                       p2_awt > 0:
                pack_awt['premisename'] = key
                pack_awt['message'] = 'The average
                   waiting time keeps on increasing.
                   Please be alert.'
                pack_awt['msg_flag'] = 'yellow'
                pack_awt['timestamp'] = time_stamp
            # 011
            if awt_flags['c_minus'] == True and
               awt_flags['c_0'] == True and \
                    awt_flags['c_plus'] == False and
                       p2_awt > 0:
                pack_awt['premisename'] = key
                pack_awt['message'] = 'The average
                   waiting time keeps on increasing
                   and passed the normal rate. Please
                   open another counter and supervise
                   the front desk officer.'
                pack_awt['msg_flag'] = 'red'
                pack_awt['timestamp'] = time_stamp
            # 100
            if awt_flags['c_minus'] == False and
               awt_flags['c_0'] == False and \
                    awt_flags['c_plus'] == True and
                       p2_awt > 0:
                pack_ast['premisename'] = key
                pack_ast['message'] = 'The average
                   waiting is far above or exceeded
                   normal rate, please open all
                   counter.  Please closely monitor
                   front desk officer.'
                pack_ast['msg_flag'] = 'red'
                pack_ast['timestamp'] = time_stamp
            # 110
            if awt_flags['c_minus'] == False and
               awt_flags['c_0'] == True and \
                    awt_flags['c_plus'] == True and
                       p2_awt > 0:
                pack_awt['premisename'] = key
                pack_awt['message'] = 'The average
                   waiting is far above or exceeded
                   normal rate, please open all
                   counter.  Please closely monitor
                   front desk officer.'
                pack_awt['msg_flag'] = 'red'
                pack_awt['timestamp'] = time_stamp
            # 111
```

```
421                         if awt_flags['c_minus'] == True and
                               awt_flags['c_0'] == True and \
422                                 awt_flags['c_plus'] == True and
                                       p2_awt > 0:
423                             pack_awt['premisename'] = key
424                             pack_awt['message'] = 'The average
                                   waiting is far above or exceeded
                                   normal rate, please open all
                                   counter.  Please closely monitor
                                   front desk officer.'
425                             pack_awt['msg_flag'] = 'red'
426                             pack_awt['timestamp'] = time_stamp
427                         # Special Case
428                         if p2_awt > plus_threshold_awt and p1_awt
                               > plus_threshold_awt:
429                             pack_awt['premisename'] = key
430                             pack_awt['message'] = 'The average
                                   waiting time keeps on increasing
                                   and passed the normal rate. Please
                                   open another counter and supervise
                                   the front desk officer.'
431                             pack_awt['msg_flag'] = 'red'
432                             pack_awt['timestamp'] = time_stamp
433                     if trend == 'down':
434                         # 001
435                         if awt_flags['c_minus'] == True and
                               awt_flags['c_0'] == False and \
436                                 awt_flags['c_plus'] == False and
                                       p2_awt > 0:
437                             pack_awt['premisename'] = key
438                             if active_counter > 3:
439                                 pack_awt['message'] = 'Great, the
                                       average waiting time is normal.
                                        You can close 2 counters'
440                             else:
441                                 pack_awt['message'] = 'Great, the
                                       average waiting time is normal.
                                       '
442                             pack_awt['msg_flag'] = 'green'
443                             pack_awt['timestamp'] = time_stamp
444                         # 010
445                         if awt_flags['c_minus'] == False and
                               awt_flags['c_0'] == True and \
446                                 awt_flags['c_plus'] == False and
                                       p2_awt > 0:
447                             pack_awt['premisename'] = key
448                             pack_awt['message'] = 'Well done, the
                                   average waiting time become normal.
                                   '
449                             pack_awt['msg_flag'] = 'green'
450                             pack_awt['timestamp'] = time_stamp
451                         # 011
452                         if awt_flags['c_minus'] == True and
                               awt_flags['c_0'] == True and \
```

```
453                     awt_flags['c_plus'] == False and
                            p2_awt > 0:
454                 pack_awt['premisename'] = key
455                 pack_awt['premisename'] = key
456                 if active_counter > 3:
457                     pack_awt['message'] = 'Great, the
                            average waiting time is normal.
                             You can close 2 counters'
458                 else:
459                     pack_awt['message'] = 'Great, the
                            average waiting time is normal.
                            '
460                 pack_awt['msg_flag'] = 'green'
461                 pack_awt['timestamp'] = time_stamp
462             # 100
463             if awt_flags['c_minus'] == False and
                    awt_flags['c_0'] == False and \
464                     awt_flags['c_plus'] == True and
                            p2_awt > 0:
465                 pack_awt['premisename'] = key
466                 pack_awt['message'] = 'Good, the
                        average waiting time is decreasing,
                         keep up the good work.'
467                 pack_awt['msg_flag'] = 'green'
468                 pack_awt['timestamp'] = time_stamp
469             # 110
470             if awt_flags['c_minus'] == False and
                    awt_flags['c_0'] == True and \
471                     awt_flags['c_plus'] == True and
                            p2_awt > 0:
472                 pack_awt['premisename'] = key
473                 pack_awt['message'] = 'Excellent, the
                        average waiting time is decreasing
                        below the normal rate.'
474                 pack_awt['msg_flag'] = 'green'
475                 pack_awt['timestamp'] = time_stamp
476             # 111
477             if awt_flags['c_minus'] == True and
                    awt_flags['c_0'] == True and \
478                     awt_flags['c_plus'] == True and
                            p2_awt > 0:
479                 pack_awt['premisename'] = key
480                 if active_counter > 3:
481                     pack_awt['message'] = 'Great, the
                            average waiting time is normal.
                             You can close 2 counters'
482                 else:
483                     pack_awt['message'] = 'Great, the
                            average waiting time is normal.
                            '
484                 pack_awt['msg_flag'] = 'green'
485                 pack_awt['timestamp'] = time_stamp
486
487             # Special Case
```

```
                    if p2_awt > plus_threshold_awt and p1_awt
                        > plus_threshold_awt:
                        pack_awt['premisename'] = key
                        pack_awt['message'] = 'The average
                            waiting time is decreasing, keep up
                             the good work.'
                        pack_awt['msg_flag'] = 'red'
                        pack_awt['timestamp'] = time_stamp
            if len(pack_awt) > 0:
                awt_message.append(pack_awt)

        # State Level (Total peforemance)
        rank_message = []
        data = pd.merge(qms_rank, premisedata, on='premisename
            ', how='left')
        state_group = data.groupby(by=data.state, as_index=
            False)
        for key, value in state_group:
            pack_rank = dict()
            pack_rank['sta_name'] = key
            pack_rank['timestamp'] = time_stamp
            group = pd.DataFrame(value)
            group_TOL = group.sort_values(by=['serveCount'],
                ascending=False)
            top_three = group_TOL.head(3)
            top_three = top_three.loc[top_three['serveCount']
                != 0.0]
            top_three = list(top_three['premisename'])
            if len(top_three) > 0:
                if len(top_three) > 1:
                    performer = 'performers'
                    article = 'are'
                else:
                    performer = 'performer'
                    article = 'is'

                top_three = ", ".join(top_three)

                pack_rank['message_top'] = 'Great! ' +
                    top_three + ' '+article+' the top ' +
                    performer +\
                     ' for the day. Keep up the good work.'
            else:
                pack_rank['message_top'] = ''
            low_three = group_TOL.tail(3)
            low_three = list(low_three['premisename'])

            def lr_diff(l, r): return list(set(l).difference(r
                ))
            diff = lr_diff(low_three, top_three)
            if len(diff) > 0:
                diff_str = ", ".join(diff)
                pack_rank['message_low'] = 'Please organize
                    outlet promotion or  open day to attract
```

```
                        more visitors at '+diff_str+'.'
531                 else:
532                     pack_rank['message_low'] = ''
533                 rank_message.append(pack_rank)
534
535             # Malaysia Level (Total peforemance)
536             rank_message_mal = dict()
537             rank_message_mal['timestamp'] = time_stamp
538             group_TOL = data.sort_values(by=['serveCount'],
                    ascending=False)
539             top_three = group_TOL.head(3)
540             top_three = top_three.loc[top_three['serveCount'] !=
                    0.0]
541             top_three = list(top_three['premisename'])
542             if len(top_three) > 0:
543                 if len(top_three) > 1:
544                     performer = 'performers'
545                     article = 'are'
546                 else:
547                     performer = 'performer'
548                     article = 'is'
549
550                 top_three = ", ".join(top_three)
551
552                 rank_message_mal['message_top'] = 'Great! ' +
                        top_three + ' '+article+' the top ' + performer
                        +\
553                     ' for the day. Keep up the good work.'
554             else:
555                 rank_message_mal['message_top'] = ''
556             low_three = group_TOL.tail(3)
557             low_three = list(low_three['premisename'])
558
559             def lr_diff(l, r): return list(set(l).difference(r))
560             diff = lr_diff(low_three, top_three)
561             if len(diff) > 0:
562                 diff_str = ", ".join(diff)
563                 rank_message_mal['message_low'] = 'Please organize
                        outlet promotion or  open day to attract more
                        visitors at '+diff_str+'.'
564             else:
565                 rank_message_mal['message_low'] = ''
566
567             postdata = dict()
568             postdata['date'] = date
569             postdata['timestamp'] = time_stamp
570             postdata['mal'] = rank_message_mal
571             postdata['sta'] = rank_message
572             postdata['tmp'] = {'AST': ast_message, 'AWT':
                    awt_message, 'VIS': vis_message}
573             postmessagedata(postdata, url, port, companyname)
574         else:
575             print('QMS component is not selected for advisory
                    module.')
```

```
576         else:
577             print('No Threshold Data.')
```

ace2/advisory/smartqueue.py

```
1  import ace2
2  import pandas as pd
3  import json
4  from datetime import datetime, timedelta
5  import time
6  import numpy as np
7  import requests as rq
8  import math
9  from pytz import timezone
10
11
12 def myround(x, base=10):
13     a = int(x)/base
14     return (base * int(a))
15
16
17 # function to create a normalized time stamp based on rounding
18 def normalizeTimeStamp(timeStamp):
19     # to remove additional 0 from the timestamp
20     end_date = timeStamp.split(" ")
21     ab1 = end_date[1]
22     ab1 = ab1.split(":")
23     hour = ab1[0][:2]
24     minutes = ab1[1][:2]
25     abc = hour+':'+minutes
26     end_date = end_date[0]+" "+abc
27     normalizedTimeStamp = datetime.strptime(end_date, "%Y/%m/%d %H:%M")
28     minute = myround(normalizedTimeStamp.minute)
29     if (minute >= 60):
30         normalizedTimeStamp = normalizedTimeStamp.replace(hour=
               normalizedTimeStamp.hour+1, minute=0)
31     else:
32         normalizedTimeStamp = normalizedTimeStamp.replace(minute=minute)
33     localtz = timezone('Asia/Kuala_Lumpur')
34     normalizedTimeStamp = localtz.localize(normalizedTimeStamp)
35     return int(normalizedTimeStamp.timestamp())
36
37
38 def getqmsthresholdata(companyname, url, port):
39     request = rq.get('http://'+url+':'+port+'/ace/api/advisory/threshold?
           company='+companyname,
40                      headers={'Content-type': 'application/json'})
41     value = request.text
42     data = json.loads(value)
43
44     if data['error'] == False:
45         data = data['Threshold']
```

```
46          data_stored = []
47          if len(data) == 0:
48              return data_stored
49          else:
50              data = data[0]['data']
51              company_data = data['components']
52              if len(company_data) > 0:
53                  for each in company_data:
54                      if each['component'] == 'qms':
55                          qms_data = dict()
56                          qms_data['component'] = 'qms'
57                          oneweek = each['oneweek']
58                          oneweek_dict = dict()
59                          oneweek_tmp = pd.io.json.json_normalize(oneweek['
                                tmp'])
60                          oneweek_sta = pd.io.json.json_normalize(oneweek['
                                sta'])
61                          oneweek_mal = pd.io.json.json_normalize(oneweek['
                                mal'])
62                          oneweek_dict['tmp'] = oneweek_tmp
63                          oneweek_dict['sta'] = oneweek_sta
64                          oneweek_dict['mal'] = oneweek_mal
65                          qms_data['oneweek'] = oneweek_dict
66
67                          onemonth = each['onemonth']
68                          onemonth_dict = dict()
69                          onemonth_tmp = pd.io.json.json_normalize(onemonth[
                                'tmp'])
70                          onemonth_sta = pd.io.json.json_normalize(onemonth[
                                'sta'])
71                          onemonth_mal = pd.io.json.json_normalize(onemonth[
                                'mal'])
72                          onemonth_dict['tmp'] = onemonth_tmp
73                          onemonth_dict['sta'] = onemonth_sta
74                          onemonth_dict['mal'] = onemonth_mal
75                          qms_data['onemonth'] = onemonth_dict
76
77                          threemonths = each['threemonths']
78                          threemonths_dict = dict()
79                          threemonths_tmp = pd.io.json.json_normalize(
                                threemonths['tmp'])
80                          threemonths_sta = pd.io.json.json_normalize(
                                threemonths['sta'])
81                          threemonths_mal = pd.io.json.json_normalize(
                                threemonths['mal'])
82                          threemonths_dict['tmp'] = threemonths_tmp
83                          threemonths_dict['sta'] = threemonths_sta
84                          threemonths_dict['mal'] = threemonths_mal
85                          qms_data['threemonths'] = threemonths_dict
86                          data_stored.append(qms_data)
87                      if each['component'] == 'wifi':
88                          wifi_data = dict()
89                          wifi_data['component'] = 'wifi'
90                          data_stored.append(wifi_data)
```

```
                    if each['component'] == 'video':
                        video_data = dict()
                        video_data['component'] = 'video'
                        data_stored.append(video_data)
                    if each['component'] == 'engage':
                        engage_data = dict()
                        engage_data['component'] = 'engage'
                        data_stored.append(engage_data)
                    if each['component'] == 'target':
                        target_data = dict()
                        target_data['component'] = 'target'
                        data_stored.append(target_data)
            return data_stored


def getqmsdatafromday(date, hour, min_, url, port):
    request = rq.get('http://'+url+':'+port+'/ace/api/v1/qms?date=' +
                     date, headers={'Content-type': 'application/json'})
    value = request.text
    qmsdata = json.loads(value)

    if qmsdata['error'] == False:
        data = qmsdata['qms']

        if len(data) == 0:
            qms_avg = pd.DataFrame()
            qms_rank = pd.DataFrame()
            return (qms_avg, qms_rank)
        else:
            timestamp = date + " " + hour + ":" + min_
            # print(timestamp)
            date_timestamp = ace2.datetime.strptime(timestamp, "%Y%m%d %H
                :%M")
            str_timestamp = date_timestamp.strftime('%s')
            qms_avg = pd.DataFrame()
            qms_rank = pd.DataFrame()

            for each in data:
                premise = each['premise']
                premise_qms = each['qmsdata']
                premise_qms = pd.io.json.json_normalize(premise_qms)
                premise_qms['premisename'] = premise
                premise_qms['time_stamp'] = premise_qms['time_stamp'].
                    apply(
                        lambda x: normalizeTimeStamp(x)).astype(int)
                premise_qms['serveAvg'] = premise_qms['serveAvg'].apply(
                        lambda x: ace2.convertTime(x))
                premise_qms['waitingAvg'] = premise_qms['waitingAvg'].
                    apply(
                        lambda x: ace2.convertTime(x))
                premise_qms['waitingCount'] = premise_qms['waitingCount'].
                    astype(float)

                data1 = premise_qms.loc[premise_qms['time_stamp']
```

```
141                                         == int(str_timestamp)-3600]  # 1
                                               hour back
142
143                 data2 = premise_qms.loc[premise_qms['time_stamp'] == int(
                        str_timestamp)]
144                 data = data1.append(data2, ignore_index=True)
145
146                 qms_avg = qms_avg.append(data, ignore_index=True)
147                 qms_rank = qms_rank.append(data2, ignore_index=True)
148
149             qms_avg.fillna(0, inplace=True)
150             qms_rank.fillna(0, inplace=True)
151
152             return (qms_avg, qms_rank)
153
154
155 def postmessagedata(data, url, port, companyname):
156     url = 'http://'+url+':'+port+'/ace/api/advisory?company='+companyname
157     headers = {'Content-type': 'application/json'}
158     r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
159     print('Post :', r.text)
160
161
162 def main(date, hour, min_):
163     source = ace2.read_json('../input.json')
164     url = source['url']
165     port = source['port']
166     companydetails = ace2.pickpremisebycom(url, port)
167     premisedata = ace2.getpremises(url, port)
168     timestamp = date + " " + hour + ":" + min_
169     # print(timestamp)
170     date_timestamp = ace2.datetime.strptime(timestamp, "%Y%m%d %H:%M")
171     time_stamp = date_timestamp.strftime('%s')
172
173     for each in companydetails:
174         print('Company: ', each['name'])
175         # TMpoint level
176         companyname = each['name']
177         data = getqmsthresholdata(companyname, url, port)
178
179         if len(data) > 0:
180             for each in data:
181                 if each['component'] == 'qms':
182
183                     oneweek = each['oneweek']
184                     onemonth = each['onemonth']
185                     threemonths = each['threemonths']
186                     threemonths_tmp = threemonths['tmp']
187                     onemonth_tmp = onemonth['tmp']
188                     oneweek_tmp = oneweek['tmp']
189
190                     qms_avg, qms_rank = getqmsdatafromday(date, hour, min_
                            , url, port)
191
```

```
                data_group = qms_avg.groupby(by=qms_avg.premisename,
                    as_index=False)

                # Tmpoint Level
                awt_message = []
                vis_message = []

                for key, value in data_group:
                    group = pd.DataFrame(value)
                    group = group.reset_index()

                    threshold_df = threemonths_tmp.loc[threemonths_tmp
                        ['premisename'] == key]

                    # TMpoint visitor
                    if int(hour) > 17:
                        oneweek_df = oneweek_tmp.loc[oneweek_tmp['
                            premisename'] == key]
                        onemonth_df = onemonth_tmp.loc[onemonth_tmp['
                            premisename'] == key]
                        threemonths_df = threemonths_tmp.loc[
                            threemonths_tmp['premisename'] == key]
                        if len(oneweek_df) > 0 and len(onemonth_df) >
                            0 and len(threemonths_df) > 0:
                            oneweek_threshold = int(oneweek_df.iloc
                                [0]['tol_serve'])
                            minus_oneweek = oneweek_threshold - (
                                oneweek_threshold * 0.25)

                            onemonth_threshold = int(onemonth_df.iloc
                                [0]['tol_serve'])
                            minus_onemonth = onemonth_threshold - (
                                onemonth_threshold * 0.25)

                            threemonths_threshold = int(threemonths_df
                                .iloc[0]['tol_serve'])
                            minus_threemonths = threemonths_threshold
                                - \
                                (threemonths_threshold * 0.25)

                            visitor = int(group.iloc[1]['serveCount'])
                            pack_vis = dict()
                            if visitor < minus_oneweek:
                                pack_vis['premisename'] = key
                                pack_vis['message'] = 'Please organise
                                    an open day to attract more
                                    visitors.'
                                pack_vis['timestamp'] = time_stamp

                            if visitor < minus_onemonth:
                                pack_vis['premisename'] = key
                                pack_vis['message'] = 'Please organize
                                    in outlet promo to attract more
                                    visitors.'
```

```
                    pack_vis['timestamp'] = time_stamp

                if visitor < minus_threemonths:
                    pack_vis['premisename'] = key
                    pack_vis['message'] = 'Please be
                        advised to relocate the outlet.'
                    pack_vis['timestamp'] = time_stamp

                if len(pack_vis) > 0:
                    vis_message.append(pack_vis)

        # AVG Waiting Time
        awt_flags = {
            'c_plus': False,
            'c_0': False,
            'c_minus': False
        }
        pack_awt = dict()

        if len(threshold_df) > 0 and len(group) == 2:
            p1_awt = group.iloc[0]['waitingAvg']
            p2_awt = group.iloc[1]['waitingAvg']
            active_counter = group.iloc[1]['activeCounter'
                ]

            trend = 'up' if p1_awt < p2_awt else 'down'

            if p1_awt == p2_awt:
                trend = 'same'
            threshold_awt = threshold_df.iloc[0]['
                waitingAvg']
            # if less than 5 min, make it avg to 5 min
            if threshold_awt < 300.0:
                threshold_awt = 300.0
            plus_threshold_awt = threshold_awt + (
                threshold_awt * 0.25)
            minus_threshold_awt = threshold_awt - (
                threshold_awt * 0.25)

            if (p2_awt >= plus_threshold_awt and
                plus_threshold_awt >= p1_awt) or \
                    (p2_awt < plus_threshold_awt and
                        plus_threshold_awt < p1_awt):
                awt_flags['c_plus'] = True

            if (p2_awt >= threshold_awt and threshold_awt
                >= p1_awt) or \
                    (p2_awt < threshold_awt and
                        threshold_awt < p1_awt):
                awt_flags['c_0'] = True

            if (p2_awt >= minus_threshold_awt and
                minus_threshold_awt >= p1_awt)or \
```

```
273                         (p2_awt < minus_threshold_awt and
                               minus_threshold_awt < p1_awt):
274                     awt_flags['c_minus'] = True
275
276                 if trend == 'up':
277                     # 001
278                     if awt_flags['c_minus'] == True and
                           awt_flags['c_0'] == False and \
279                             awt_flags['c_plus'] == False and
                                   p2_awt > 0:
280                         pack_awt['premisename'] = key
281                         pack_awt['message'] = 'The average
                               waiting time is approaching\
282                                                 waiting time
                                                       limit,
                                                       please be
                                                       alert.'
283                         pack_awt['msg_flag'] = 'yellow'
284                         pack_awt['timestamp'] = time_stamp
285
286                     # 011
287                     if awt_flags['c_minus'] == True and
                           awt_flags['c_0'] == True and \
288                             awt_flags['c_plus'] == False and
                                   p2_awt > 0:
289                         pack_awt['premisename'] = key
290                         pack_awt['message'] = 'The average
                               waiting time is crossing the
                               waiting time limit.\
291                                                 Please be alert
                                                       and you are
                                                       adviced to
                                                      open another
                                                       counter.'
292                         pack_awt['msg_flag'] = 'red'
293                         pack_awt['timestamp'] = time_stamp
294
295                     # 111
296                     if awt_flags['c_minus'] == True and
                           awt_flags['c_0'] == True and \
297                             awt_flags['c_plus'] == True and
                                   p2_awt > 0:
298                         pack_awt['premisename'] = key
299                         pack_awt['message'] = 'The average
                               waiting time is far above waiting
                               time limit,\
300                                                 please open all
                                                       counter.'
301                         pack_awt['msg_flag'] = 'red'
302                         pack_awt['timestamp'] = time_stamp
303                     # Special Case
304                     if p2_awt > plus_threshold_awt and p1_awt
                           > plus_threshold_awt:
305                         pack_awt['premisename'] = key
```

```
306                                 pack_awt['message'] = 'The average
                                        waiting time is far above waiting
                                        time limit,\
307                                                                 please open all
                                                                        counter.'
308                                 pack_awt['msg_flag'] = 'red'
309                                 pack_awt['timestamp'] = time_stamp
310                             if trend == 'down':
311                                 # 001
312                                 if awt_flags['c_minus'] == True and
                                        awt_flags['c_0'] == False and \
313                                          awt_flags['c_plus'] == False and
                                             p2_awt > 0:
314                                 pack_awt['premisename'] = key
315                                 pack_awt['message'] = 'Great, the
                                        average waiting time become normal.
                                        '
316                                 pack_awt['msg_flag'] = 'green'
317                                 pack_awt['timestamp'] = time_stamp
318                                 # 010
319                                 if awt_flags['c_minus'] == False and
                                        awt_flags['c_0'] == True and \
320                                          awt_flags['c_plus'] == False and
                                             p2_awt > 0:
321                                 pack_awt['premisename'] = key
322                                 pack_awt['message'] = 'Great, the
                                        average waiting time become normal.
                                        '
323                                 pack_awt['msg_flag'] = 'green'
324                                 pack_awt['timestamp'] = time_stamp
325                                 # 011
326                                 if awt_flags['c_minus'] == True and
                                        awt_flags['c_0'] == True and \
327                                          awt_flags['c_plus'] == False and
                                             p2_awt > 0:
328                                 pack_awt['premisename'] = key
329                                 pack_awt['message'] = 'Great, the
                                        average waiting time become normal.
                                        '
330                                 pack_awt['msg_flag'] = 'green'
331                                 pack_awt['timestamp'] = time_stamp
332                                 # 100
333                                 if awt_flags['c_minus'] == False and
                                        awt_flags['c_0'] == False and \
334                                          awt_flags['c_plus'] == True and
                                             p2_awt > 0:
335                                 pack_awt['premisename'] = key
336                                 pack_awt['message'] = 'Great, the
                                        average waiting time become normal.
                                        '
337                                 pack_awt['msg_flag'] = 'green'
338                                 pack_awt['timestamp'] = time_stamp
339                                 # 110
```

```
                    if awt_flags['c_minus'] == False and
                       awt_flags['c_0'] == True and \
                            awt_flags['c_plus'] == True and
                               p2_awt > 0:
                        pack_awt['premisename'] = key
                        pack_awt['message'] = 'Great, the
                           average waiting time become normal.
                           '
                        pack_awt['msg_flag'] = 'green'
                        pack_awt['timestamp'] = time_stamp
                    # 111
                    if awt_flags['c_minus'] == True and
                       awt_flags['c_0'] == True and \
                            awt_flags['c_plus'] == True and
                               p2_awt > 0:
                        pack_awt['premisename'] = key
                        if active_counter > 3:
                            pack_awt['message'] = 'Great, the
                               average waiting time is normal.
                                \
                                                    You can
                                                       remain 2

                                                       counters
                                                        open.'
                        else:
                            pack_awt['message'] = 'Great, the
                               average waiting time is normal.
                               '
                        pack_awt['msg_flag'] = 'green'
                        pack_awt['timestamp'] = time_stamp

            if len(pack_awt) > 0:
                awt_message.append(pack_awt)

        # State Level (Total peforemance)
        rank_message = []
        data = pd.merge(qms_rank, premisedata, on='premisename
           ', how='left')
        state_group = data.groupby(by=data.state, as_index=
           False)
        for key, value in state_group:
            pack_rank = dict()
            pack_rank['sta_name'] = key
            pack_rank['timestamp'] = time_stamp
            group = pd.DataFrame(value)
            group_TOL = group.sort_values(by=['serveCount'],
               ascending=False)
            top_three = group_TOL.head(3)
            top_three = top_three.loc[top_three['serveCount']
               != 0.0]
            top_three = list(top_three['premisename'])
            if len(top_three) > 0:
                if len(top_three) > 1:
```

```
                        performer = 'performers'
                        article = 'are'
                    else:
                        performer = 'performer'
                        article = 'is'

                    top_three = ", ".join(top_three)

                    pack_rank['message_top'] = 'Great! ' +
                       top_three + ' '+article+' the top ' +
                       performer +\
                        ' for the day. Keep up the good work.'
                else:
                    pack_rank['message_top'] = ''
                low_three = group_TOL.tail(3)
                low_three = list(low_three['premisename'])

                def lr_diff(l, r): return list(set(l).difference(r
                   ))
                diff = lr_diff(low_three, top_three)
                if len(diff) > 0:
                    diff_str = ", ".join(diff)
                    pack_rank['message_low'] = 'Please organize
                       outlet promotion or  open day to attract
                       more visitors at '+diff_str+'.'
                else:
                    pack_rank['message_low'] = ''
                rank_message.append(pack_rank)

            # Malaysia Level (Total peforemance)
            rank_message_mal = dict()
            rank_message_mal['timestamp'] = time_stamp
            group_TOL = data.sort_values(by=['serveCount'],
               ascending=False)
            top_three = group_TOL.head(3)

            top_three = top_three.loc[top_three['serveCount'] !=
               0.0]
            top_three = list(top_three['premisename'])
            if len(top_three) > 0:
                if len(top_three) > 1:
                    performer = 'performers'
                    article = 'are'
                else:
                    performer = 'performer'
                    article = 'is'

                top_three = ", ".join(top_three)

                rank_message_mal['message_top'] = 'Great! ' +
                   top_three + ' '+article+' the top ' + performer
                    +\
                    ' for the day. Keep up the good work.'
            else:
```

```
                        rank_message_mal['message_top'] = ''
                    low_three = group_TOL.tail(3)
                    low_three = list(low_three['premisename'])

                    def lr_diff(l, r): return list(set(l).difference(r))
                    diff = lr_diff(low_three, top_three)
                    if len(diff) > 0:
                        diff_str = ", ".join(diff)
                        rank_message_mal['message_low'] = 'Please organize
                            outlet promotion or  open day to attract more
                           visitors at '+diff_str+'.'
                    else:
                        rank_message_mal['message_low'] = ''

                    postdata = dict()
                    postdata['date'] = date
                    postdata['timestamp'] = time_stamp
                    postdata['mal'] = rank_message_mal
                    postdata['sta'] = rank_message
                    postdata['tmp'] = {'AWT': awt_message, 'VIS':
                       vis_message}
                    postmessagedata(postdata, url, port, companyname)
                else:
                    print(each['component'], 'component is not selected
                       for advisory module.')
        else:
            print('No Threshold Data.')
```

ace2/advisory/threshold.py

```
import ace2
import pandas as pd
import json
import requests as rq


def getqmspastdata(companyname,url,port,path):
    request = rq.get ('http://'+url+':'+port+'/ace/api/advisory/'+path+'?
       company='+companyname, \
            headers= {'Content-type': 'application/json'})
    value = request.text
    data = json.loads(value)
    if data['error'] == False:
        if path == 'oneweek':
            data = data['QMS7DaysData']
        elif path == 'onemonth':
            data = data['QMS1MonthData']
        elif path == 'threemonths':
            data = data['QMS3MonthlyData']
        else:
            tmp_data = pd.DataFrame()
            sta_data = pd.DataFrame()
```

```
22              mal_data = pd.DataFrame()
23              return (mal_data,sta_data,tmp_data)
24
25          if len(data) == 0:
26              tmp_data = pd.DataFrame()
27              sta_data = pd.DataFrame()
28              mal_data = pd.DataFrame()
29              return (mal_data,sta_data,tmp_data)
30          else:
31              columns_flag = False
32              for each in data:
33                  dailydata = each['data']
34                  dailytmpoints = dailydata['tmpoint_detail']
35                  dailystates = dailydata['state_detail']
36                  dailystates = pd.io.json.json_normalize(dailystates)
37                  dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
38                  dailymal = pd.io.json.json_normalize(dailydata)
39                  dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                        inplace = True)
40                  if columns_flag == False:
41                      tmp_data = dailytmpoints
42                      sta_data = dailystates
43                      mal_data = dailymal
44                      columns_flag= True
45                  else:
46
47                      tmp_data = tmp_data.append(dailytmpoints, ignore_index
                            =True)
48                      sta_data = sta_data.append(dailystates, ignore_index=
                            True)
49                      mal_data = mal_data.append(dailymal, ignore_index=True
                            )
50              return (mal_data,sta_data,tmp_data)
51
52  # post Operations
53  def postthreshold(data,url,port,companyname):
54      url = 'http://'+url+':'+port+'/ace/api/advisory/threshold?company='+
            companyname
55      headers = {'Content-type': 'application/json'}
56      r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
57      print('Post postthreshold:',r.text)
58  def thresholdcal(companyname,url,port,path):
59      mal_data,sta_data,tmp_data = getqmspastdata(companyname,url,port,path)
60
61      # TMpoint Levels
62      threshold_tmpoint = []
63      if len(tmp_data) >0:
64          group_data = tmp_data.groupby(by=tmp_data.premisename,as_index=
                False)
65
66          for key,value in group_data:
67              stored_data = dict()
68              group = pd.DataFrame(value)
```

```
69              group['serveAvg'] = group['serveAvg'].apply(lambda x: ace2.
                    Mintoseconds(x))
70              group['waitingAvg'] = group['waitingAvg'].apply(lambda x: ace2
                    .Mintoseconds(x))
71
72              group_AST = group.sort_values(by=['serveAvg'])
73              group_AWT = group.sort_values(by=['waitingAvg'])
74              group_TOL = group.sort_values(by=['tol_serve'])
75
76              stored_data['premisename'] = group_AST.iloc[0]['premisename']
77              stored_data['state'] = group_AST.iloc[0]['state']
78              stored_data['serveAvg'] =group_AST['serveAvg'].median()
79              stored_data['waitingAvg'] =group_AWT['waitingAvg'].median()
80              stored_data['tol_serve'] = round(group_TOL['tol_serve'].mean()
                    )
81              threshold_tmpoint.append(stored_data)
82
83      # State Level
84      threshold_state = []
85      if len(sta_data)>0:
86          sta_data = sta_data[['sta_name','sta_tol_ser','sta_avg_ast','
                sta_avg_awt']]
87          # sta_data['sta_avg_ast'] = sta_data['sta_avg_ast'].apply(lambda x
                : ace2.Mintoseconds(x))
88          # sta_data['sta_avg_awt'] = sta_data['sta_avg_awt'].apply(lambda x
                : ace2.Mintoseconds(x))
89          sta_group_data = sta_data.groupby(by=sta_data.sta_name,as_index=
                False)
90          threshold_sta_data = []
91          for key,value in sta_group_data:
92              stored_data = dict()
93              group = pd.DataFrame(value)
94
95              # group_AST = group.sort_values(by=['sta_avg_ast'])
96              # group_AWT = group.sort_values(by=['sta_avg_awt'])
97              group_TOL = group.sort_values(by=['sta_tol_ser'])
98
99              stored_data['sta_name'] = group_TOL.iloc[0]['sta_name']
100             # stored_data['sta_avg_ast'] =group_AST['sta_avg_ast'].median
                    ()
101             # stored_data['sta_avg_awt'] =group_AWT['sta_avg_awt'].median
                    ()
102             stored_data['sta_tol_ser'] = round(group_TOL['sta_tol_ser'].
                    mean())
103             threshold_state.append(stored_data)
104
105     # Malaysia Level
106     threshold_mal = []
107     if len(mal_data) > 0:
108         mal_data = mal_data[['mal_avg_ast','mal_avg_awt','mal_tol_ser']]
109         # mal_data['mal_avg_ast'] = mal_data['mal_avg_ast'].apply(lambda x
                : ace2.Mintoseconds(x))
110         # mal_data['mal_avg_awt'] = mal_data['mal_avg_awt'].apply(lambda x
                : ace2.Mintoseconds(x))
```

```
111         mal_data['mal_tol_ser'] = mal_data['mal_tol_ser'].astype(int)
112
113
114         # group_AST = mal_data.sort_values(by=['mal_avg_ast'])
115         # group_AWT = mal_data.sort_values(by=['mal_avg_awt'])
116         group_TOL = mal_data.sort_values(by=['mal_tol_ser'])
117
118         stored_data = dict()
119             # stored_data['mal_avg_ast'] =group_AST['mal_avg_ast'].median
                ()
120             # stored_data['mal_avg_awt'] =group_AWT['mal_avg_awt'].median
                ()
121         stored_data['mal_tol_ser'] = round(group_TOL['mal_tol_ser'].mean()
            )
122         threshold_mal.append(stored_data)
123     data = dict()
124     data['tmp'] = threshold_tmpoint
125     data['sta'] = threshold_state
126     data['mal'] = threshold_mal
127     return data
128
129 def main(date):
130     source = ace2.read_json('../input.json')
131     url = source['url']
132     port = source['port']
133     companydetails = ace2.pickpremisebycom(url,port)
134     for each in companydetails:
135         post_data = dict()
136         post_data['date'] = date
137         post_data['components'] = []
138
139         # for QMS
140         data_qms = dict()
141         data_qms['component'] = 'qms'
142         # companyname = 'TSSSB'
143         companyname = each['name']
144         # A Week threshold
145         path = 'oneweek'
146         # thresholdcal(companyname,url,port,path)
147         threshold_week = thresholdcal(companyname,url,port,path)
148         data_qms['oneweek'] = threshold_week
149         # Month threshold
150         path = 'onemonth'
151         threshold_month = thresholdcal(companyname,url,port,path)
152         data_qms['onemonth'] = threshold_month
153         # Three months threshold
154         path = 'threemonths'
155         threshold_three_months = thresholdcal(companyname,url,port,path)
156         data_qms['threemonths'] = threshold_three_months
157         post_data['components'].append(data_qms)
158
159         # for wifi
160         data_wifi = dict()
161         data_wifi['component'] = 'wifi'
```

```
162         post_data['components'].append(data_wifi)
163
164         if len(threshold_week['tmp']) > 0 and  len(threshold_month['tmp'])
                > 0 and\
165         len(threshold_three_months['tmp']) > 0 :
166             postthreshold(post_data,url,port,companyname)
```

ace2/dwell/daily.py

```
1  import requests as rq
2  import json
3  import ace2
4  import pandas as pd
5
6
7  def min_seconds(time_str):
8      m, s = time_str.split(':')
9      return float(m) * 60 + float(s)
10
11
12 def seconds_mins(s):
13     if math.isnan(s) == True:
14         return ("00:00")
15     else:
16         m, s = divmod(s, 60)
17         h, m = divmod(m, 60)
18         return ("%02d:%02d" % (m, s))
19
20
21 def getdwelldatadaily(date, url, port):
22
23     request = rq.get('http://'+url+':'+port+'/ace/api/v2/dwell?date='+date
           ,
24                      headers={'Content-type': 'application/json'})
25     value = request.text
26     dwelldata = json.loads(value)
27     error = True
28     dwell_final = pd.DataFrame()
29     if dwelldata['error'] == False:
30         data = dwelldata['dwellData']
31         for each in data:
32             dwell = pd.DataFrame(each['data1'])
33             dwell['premisename'] = each['premise_name']
34             dwell['dwell_time'] = dwell['dwell_time'].apply(lambda x:
                   min_seconds(x))
35
36             dwell_final = dwell_final.append(dwell, ignore_index=True)
37         error = False
38         return(dwell_final, error)
39     else:
40         return(dwell_final, error)
41
```

```
42
43 seconds_mins
44
45
46 def postdwell(mal_detail, url, port, company):
47     url = 'http://'+url+':'+port+'/ace/api/v2/dwell/daily?company='+
          company
48     headers = {'Content-type': 'application/json'}
49     r = rq.post(url, data=json.dumps(mal_detail), headers=headers)
50     return(r.text)
51
52
53 def dwelloperation(data, date, timestamp):
54
55     # TMpoint  Operations
56     # Group by TMpoint Name
57     tmp_group = data.groupby(by=data.premisename, as_index=False)
58     tmpoint_detail = list()
59     for key, value in tmp_group:
60         group = pd.DataFrame(value)
61         tmp = dict()
62         tmp['premisename'] = group.iloc[0]['premisename']
63         tmp['state'] = group.iloc[0]['state']
64
65         # Group by advert_title
66         adv_group = group.groupby(by=group.advert_title, as_index=False)
67         adv_detail = list()
68         for key, each in adv_group:
69             adv = pd.DataFrame(each)
70             adv_each = dict()
71             adv_each['advert_title'] = adv.iloc[0]['advert_title']
72             adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
73             adv_detail.append(adv_each)
74
75         tmp['adv_detail'] = adv_detail
76         tmpoint_detail.append(tmp)
77     # print(tmpoint_detail)
78
79     # State Operations
80     sta_group = data.groupby(by=data.state, as_index=False)
81     state_detail = list()
82     for key, value in sta_group:
83         group = pd.DataFrame(value)
84         sta = dict()
85         sta['sta_name'] = group.iloc[0]['state']
86         # Group by advert_title
87         adv_group = group.groupby(by=group.advert_title, as_index=False)
88         adv_detail = list()
89         for key, each in adv_group:
90             adv = pd.DataFrame(each)
91             adv_each = dict()
92             adv_each['advert_title'] = adv.iloc[0]['advert_title']
93             adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
94             adv_detail.append(adv_each)
```

```
95
96          sta['adv_detail'] = adv_detail
97          state_detail.append(sta)
98      # print(state_detail)
99
100     # Malaysia Operations
101     mal_detail = dict()
102     mal_detail['date'] = date
103     mal_detail['timestamp'] = timestamp
104     mal_group = data.groupby(by=data.advert_title, as_index=False)
105     mal_adv_detail = list()
106     for key, each in mal_group:
107         adv = pd.DataFrame(each)
108         adv_each = dict()
109         adv_each['advert_title'] = adv.iloc[0]['advert_title']
110         adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
111
112         mal_adv_detail.append(adv_each)
113     mal_detail['mal_detail'] = mal_adv_detail
114     mal_detail['state_detail'] = state_detail
115     mal_detail['tmpoint_detail'] = tmpoint_detail
116     # print(mal_detail)
117     return (mal_detail)
118
119
120 def main(date, hour, min_, loop=False):
121     source = ace2.read_json('../input.json')
122     url = source['url']
123     port = source['port']
124     timestamp = date[0:4]+'/'+date[4:6]+'/'+date[6:]+' '+hour+':'+min_
125     print(timestamp)
126     timestamp = ace2.datetime.strptime(timestamp, "%Y/%m/%d %H:%M")
127     timestamp = timestamp.timestamp()
128     data, error = getdwelldatadaily(date, url, port)
129     if error == False:
130         premisedata = ace2.getpremises(url, port)
131
132         data = pd.merge(data, premisedata, on='premisename', how='left')
133         companydetails = ace2.pickpremisebycom(url, port)
134
135         for each in companydetails:
136             data_each = data[data['premisename'].isin(each['engagepremises
                ')]]
137             if len(data_each) == 0:
138                 print('No Data Related to ', each['name'])
139             else:
140
141                 mal_detail = dwelloperation(data, date, timestamp)
142                 resutl_text = postdwell(mal_detail, url, port, each['name'
                    ])
143                 print(each['name'], resutl_text)
```

ace2/dwell/___init___.py

```
__all__ = ['daily','weekly','monthly',]

import ace2.dwell.daily
import ace2.dwell.weekly
import ace2.dwell.monthly
```

ace2/dwell/monthly.py

```
import requests as rq
import json
import ace2
import pandas as pd

def getweeklydata(url,port,companyname,month):
    request = rq.get ('http://'+url+':'+port+'/ace/api/v2/dwell/monthlydata?company='\
        +companyname+'&month='+month, headers= {'Content-type': 'application/json'})

    value = request.text
    predata = json.loads(value)

    error = True

    if predata['error'] == False:
        data = predata['DwellMonthlylyData']


        if len(data) > 0:
            tm_pre_final = ace2.pd.DataFrame()
            for each in data:
                data_each = each['data']
                date = data_each['date']
                tm_predata = ace2.pd.io.json.json_normalize(data_each['tmpoint_detail'])
                for each in range(len(tm_predata['adv_detail'])):
                    tm_each = tm_predata['adv_detail'].iloc[each]
                    tm_each = ace2.pd.io.json.json_normalize(tm_each)
                    tm_each['premisename'] = tm_predata['premisename'].iloc[each]
                    tm_each['state'] = tm_predata['state'].iloc[each]
                    tm_each['date'] = date
                    tm_pre_final =tm_pre_final.append(tm_each,ignore_index=True)
            return(tm_pre_final)

        else:
            tm_pre_final = ace2.pd.DataFrame()
            return(tm_pre_final)
def dwelloperation(data,month):
```

```
    year = ace2.datetime.today().strftime("%Y")
    unique_key = year+str(month)
    # TMpoint  Operations
    # Group by TMpoint Name
    tmp_group = data.groupby(by=data.premisename,as_index=False)
    tmpoint_detail = list()
    for key,value in tmp_group:
        group = pd.DataFrame(value)
        tmp = dict()
        tmp['premisename'] = group.iloc[0]['premisename']
        tmp['state'] = group.iloc[0]['state']

        # Group by advert_title
        adv_group = group.groupby(by=group.advert_title,as_index=False)
        adv_detail = list()
        for key,each in adv_group:
            adv = pd.DataFrame(each)
            adv_each = dict()
            adv_each['advert_title'] = adv.iloc[0]['advert_title']
            adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
            adv_detail.append(adv_each)

        tmp['adv_detail'] = adv_detail
        tmpoint_detail.append(tmp)
    # print(tmpoint_detail)

    # State Operations
    sta_group = data.groupby(by=data.state,as_index=False)
    state_detail = list()
    for key,value in sta_group:
        group = pd.DataFrame(value)
        sta = dict()
        sta['sta_name'] = group.iloc[0]['state']
        # Group by advert_title
        adv_group = group.groupby(by=group.advert_title,as_index=False)
        adv_detail = list()
        for key,each in adv_group:
            adv = pd.DataFrame(each)
            adv_each = dict()
            adv_each['advert_title'] = adv.iloc[0]['advert_title']
            adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
            adv_detail.append(adv_each)

        sta['adv_detail'] = adv_detail
        state_detail.append(sta)
    # print(state_detail)

    # Malaysia Operations
    mal_detail = dict()
    mal_detail['date'] = unique_key

    mal_group = data.groupby(by=data.advert_title,as_index=False)
    mal_adv_detail = list()
    for key,each in mal_group:
```

```
92          adv = pd.DataFrame(each)
93          adv_each = dict()
94          adv_each['advert_title'] = adv.iloc[0]['advert_title']
95          adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
96
97          mal_adv_detail.append(adv_each)
98      mal_detail['mal_detail'] = mal_adv_detail
99      mal_detail['state_detail'] = state_detail
100     mal_detail['tmpoint_detail'] = tmpoint_detail
101     # print(mal_detail)
102     return  (mal_detail,unique_key)
103 def postdwell(mal_detail,url,port,company):
104     url = 'http://'+url+':'+port+'/ace/api/v2/dwell/monthly?company='+
            company
105     headers = {'Content-type': 'application/json'}
106     r = rq.post(url, data=json.dumps(mal_detail), headers=headers)
107     return(r.text)
108
109
110 def main(weekno):
111     source = ace2.read_json('../input.json')
112     url = source['url']
113     port = source['port']
114
115     companydetails = ace2.pickpremisebycom(url,port)
116     for each in companydetails:
117         tm_pre_final = getweeklydata(url,port,each['name'],weekno)
118         # print(tm_pre_final)
119         if len(tm_pre_final) == 0:
120             year = ace2.datetime.today().strftime("%Y")
121             unique_key = year+'-'+str(weekno)
122             print(unique_key,each['name'],'No Weekly Data for week no.',
                    weekno)
123         else:
124             mal_detail,unique_key = dwelloperation(tm_pre_final,weekno)
125             text_result = postdwell(mal_detail,url,port,each['name'])
126             print(unique_key,each['name'],text_result)
```

ace2/dwell/weekly.py

```
1  import requests as rq
2  import json
3  import ace2
4  import pandas as pd
5
6  def getweeklydata(url,port,companyname,weekno):
7      request = rq.get ('http://'+url+':'+port+'/ace/api/v2/dwell/weeklydata
           ?company='\
8              +companyname+'&week='+weekno, headers= {'Content-type': '
                   application/json'})
9
10     value = request.text
```

```
    predata = json.loads(value)

    error = True

    if predata['error'] == False:
        data = predata['DwellWeeklyData']


        if len(data) > 0:
            tm_pre_final = ace2.pd.DataFrame()
            for each in data:
                data_each = each['data']
                date = data_each['date']
                tm_predata = ace2.pd.io.json.json_normalize(data_each['
                    tmpoint_detail'])
                for each in range(len(tm_predata['adv_detail'])):
                    tm_each = tm_predata['adv_detail'].iloc[each]
                    tm_each = ace2.pd.io.json.json_normalize(tm_each)
                    tm_each['premisename'] = tm_predata['premisename'].
                        iloc[each]
                    tm_each['state'] = tm_predata['state'].iloc[each]
                    tm_each['date'] = date
                    tm_pre_final =tm_pre_final.append(tm_each,ignore_index
                        =True)


            return(tm_pre_final)

        else:
            tm_pre_final = ace2.pd.DataFrame()
            return(tm_pre_final)
def dwelloperation(data,weekno):
    year = ace2.datetime.today().strftime("%Y")
    unique_key = year+'-'+str(weekno)
    # TMpoint  Operations
    # Group by TMpoint Name
    tmp_group = data.groupby(by=data.premisename,as_index=False)
    tmpoint_detail = list()
    for key,value in tmp_group:
        group = pd.DataFrame(value)
        tmp = dict()
        tmp['premisename'] = group.iloc[0]['premisename']
        tmp['state'] = group.iloc[0]['state']

        # Group by advert_title
        adv_group = group.groupby(by=group.advert_title,as_index=False)
        adv_detail = list()
        for key,each in adv_group:
            adv = pd.DataFrame(each)
            adv_each = dict()
            adv_each['advert_title'] = adv.iloc[0]['advert_title']
            adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
            adv_detail.append(adv_each)

```

```
62          tmp['adv_detail'] = adv_detail
63          tmpoint_detail.append(tmp)
64      # print(tmpoint_detail)
65
66      # State Operations
67      sta_group = data.groupby(by=data.state,as_index=False)
68      state_detail = list()
69      for key,value in sta_group:
70          group = pd.DataFrame(value)
71          sta = dict()
72          sta['sta_name'] = group.iloc[0]['state']
73          # Group by advert_title
74          adv_group = group.groupby(by=group.advert_title,as_index=False)
75          adv_detail = list()
76          for key,each in adv_group:
77              adv = pd.DataFrame(each)
78              adv_each = dict()
79              adv_each['advert_title'] = adv.iloc[0]['advert_title']
80              adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
81              adv_detail.append(adv_each)
82
83          sta['adv_detail'] = adv_detail
84          state_detail.append(sta)
85      # print(state_detail)
86
87      # Malaysia Operations
88      mal_detail = dict()
89      mal_detail['date'] = unique_key
90
91      mal_group = data.groupby(by=data.advert_title,as_index=False)
92      mal_adv_detail = list()
93      for key,each in mal_group:
94          adv = pd.DataFrame(each)
95          adv_each = dict()
96          adv_each['advert_title'] = adv.iloc[0]['advert_title']
97          adv_each['dwell_time'] = max(adv['dwell_time'].astype(int))
98
99          mal_adv_detail.append(adv_each)
100     mal_detail['mal_detail'] = mal_adv_detail
101     mal_detail['state_detail'] = state_detail
102     mal_detail['tmpoint_detail'] = tmpoint_detail
103     # print(mal_detail)
104     return  (mal_detail,unique_key)
105 def postdwell(mal_detail,url,port,company):
106     url = 'http://'+url+':'+port+'/ace/api/v2/dwell/weekly?company='+
            company
107     headers = {'Content-type': 'application/json'}
108     r = rq.post(url, data=json.dumps(mal_detail), headers=headers)
109     return(r.text)
110
111
112 def main(weekno):
113     source = ace2.read_json('../input.json')
114     url = source['url']
```

```
    port = source['port']

    companydetails = ace2.pickpremisebycom(url,port)
    for each in companydetails:
        tm_pre_final = getweeklydata(url,port,each['name'],weekno)
        # print(tm_pre_final)
        if len(tm_pre_final) == 0:
            year = ace2.datetime.today().strftime("%Y")
            unique_key = year+'-'+str(weekno)
            print(unique_key,each['name'],'No Weekly Data for week no.',
                weekno)
        else:
            mal_detail,unique_key = dwelloperation(tm_pre_final,weekno)
            text_result = postdwell(mal_detail,url,port,each['name'])
            print(unique_key,each['name'],text_result)
```

ace2/emo/daily.py

```
import requests as rq
import json
import ace2
import pandas as pd

def getemodatadaily(date,url,port,path,timestamp):
    request = rq.get('http://'+url+':'+port+'/ace/api/v2/engage/'+path+'?
        date='+date, \
         headers= {'Content-type': 'application/json'})
    value = request.text
    emodata = json.loads(value)
    error = True
    df_final = pd.DataFrame()

    if emodata['error'] == False:
        if path == 'last5min':
            data = emodata['engageLast5minData']
        else:
            data = emodata['EngagementData']
        for each in data:
            if path == 'last5min':
                gender = pd.DataFrame([each['gender_data']])
                gender['premisename'] = each['premise_name']
                emotion = pd.DataFrame([each['emotion_data']])
                emotion['premisename'] = each['premise_name']

            else:
                gender = pd.DataFrame(each['gender_data'])
                gender['premisename'] = each['premise_name']
                emotion = pd.DataFrame(each['emotion_data'])
                emotion['premisename'] = each['premise_name']
            keys = ['premisename','advert_title','time_stamp']
            df_new =  pd.merge(emotion, gender, on=keys, how='left')
            df_final = df_final.append(df_new,ignore_index=True)
```

```
34
35          df_final['time_stamp'] = df_final['time_stamp'].apply(lambda x:
                ace2.normalizeTimeStamp(x))
36          # df_final = df_final.loc[df_final['time_stamp'] == timestamp]
37          error = False
38          return(df_final,error)
39      else:
40          return(df_final,error)
41  def getpreviousdata(date,url,port,companyname):
42      request = rq.get ('http://'+url+':'+port+'/ace/api/v2/emotion/lastdata
            ?date='+date+\
43          '&company='+companyname, headers= {'Content-type': 'application/
                json'})
44
45      value = request.text
46      predata = json.loads(value)
47      error = True
48      if predata['error'] == False:
49          data = predata['EmotionLastData']
50          if len(data) > 0:
51
52              data = data[0]['data']
53              timestamp_pre = data['timestamp']
54              mal_pre_final = ace2.pd.io.json.json_normalize(data['
                    mal_detail'])
55
56              sta_pre_final = ace2.pd.DataFrame()
57              sta_predata = ace2.pd.io.json.json_normalize(data['
                    state_detail'])
58              for each in range(len(sta_predata['adv_detail'])):
59                  sta_each = sta_predata['adv_detail'].iloc[each]
60                  sta_each = ace2.pd.io.json.json_normalize(sta_each)
61                  sta_each['sta_name'] = sta_predata['sta_name'].iloc[each]
62                  sta_pre_final =sta_pre_final.append(sta_each,ignore_index=
                        True)
63
64              tm_pre_final = ace2.pd.DataFrame()
65              tm_predata = ace2.pd.io.json.json_normalize(data['
                    tmpoint_detail'])
66              for each in range(len(tm_predata['adv_detail'])):
67                  tm_each = tm_predata['adv_detail'].iloc[each]
68                  tm_each = ace2.pd.io.json.json_normalize(tm_each)
69                  tm_each['premisename'] = tm_predata['premisename'].iloc[
                        each]
70                  tm_each['state'] = tm_predata['state'].iloc[each]
71                  tm_each['time_stamp'] = timestamp_pre
72                  tm_pre_final =tm_pre_final.append(tm_each,ignore_index=
                        True)
73
74              return(timestamp_pre,mal_pre_final,sta_pre_final,tm_pre_final)
75          else:
76              timestamp_pre = ''
77              mal_pre_final = ace2.pd.DataFrame()
78              sta_pre_final = ace2.pd.DataFrame()
```

```
79              tm_pre_final = ace2.pd.DataFrame()
80              return(timestamp_pre,mal_pre_final,sta_pre_final,tm_pre_final)
81
82
83  def postemo(mal_detail,url,port,company):
84      url = 'http://'+url+':'+port+'/ace/api/v2/emotion/daily?company='+
            company
85      headers = {'Content-type': 'application/json'}
86      r = rq.post(url, data=json.dumps(mal_detail), headers=headers)
87      print(r.text)
88
89  def normalOps(data,date,timestamp):
90      # TMpoint  Operations
91      # Group by TMpoint Name
92      tmp_group = data.groupby(by=data.premisename,as_index=False)
93      tmpoint_detail = list()
94      for key,value in tmp_group:
95          group = pd.DataFrame(value)
96          tmp = dict()
97          tmp['premisename'] = group.iloc[0]['premisename']
98          tmp['state'] = group.iloc[0]['state']
99
100         # Group by advert_title
101         adv_group = group.groupby(by=group.advert_title,as_index=False)
102         adv_detail = list()
103         for key,each in adv_group:
104             adv = pd.DataFrame(each)
105             adv_each = dict()
106             adv_each['advert_title'] = adv.iloc[0]['advert_title']
107             adv_each['male'] = sum(adv['male'].astype(int))
108             adv_each['female'] = sum(adv['female'].astype(int))
109             adv_each['interested'] = sum(adv['interested'].astype(int))
110             adv_each['neutral'] = sum(adv['neutral'].astype(int))
111             adv_each['notinterested'] = sum(adv['notinterested'].astype(
                    int))
112             adv_detail.append(adv_each)
113
114         tmp['adv_detail'] = adv_detail
115         tmpoint_detail.append(tmp)
116     # print(tmpoint_detail)
117
118     # State Operations
119     sta_group = data.groupby(by=data.state,as_index=False)
120     state_detail = list()
121     for key,value in sta_group:
122         group = pd.DataFrame(value)
123         sta = dict()
124         sta['sta_name'] = group.iloc[0]['state']
125         # Group by advert_title
126         adv_group = group.groupby(by=group.advert_title,as_index=False)
127         adv_detail = list()
128         for key,each in adv_group:
129             adv = pd.DataFrame(each)
130             adv_each = dict()
```

```
131                 adv_each['advert_title'] = adv.iloc[0]['advert_title']
132                 adv_each['male'] = sum(adv['male'].astype(int))
133                 adv_each['female'] = sum(adv['female'].astype(int))
134                 adv_each['interested'] = sum(adv['interested'].astype(int))
135                 adv_each['neutral'] = sum(adv['neutral'].astype(int))
136                 adv_each['notinterested'] = sum(adv['notinterested'].astype(
                        int))
137                 adv_detail.append(adv_each)
138
139             sta['adv_detail'] = adv_detail
140             state_detail.append(sta)
141         # print(state_detail)
142
143         # Malaysia Operations
144         mal_detail = dict()
145         mal_detail['date'] = date
146         mal_detail['timestamp'] = timestamp
147         mal_group = data.groupby(by=data.advert_title,as_index=False)
148         mal_adv_detail = list()
149         for key,each in mal_group:
150             adv = pd.DataFrame(each)
151             adv_each = dict()
152             adv_each['advert_title'] = adv.iloc[0]['advert_title']
153             adv_each['male'] = sum(adv['male'].astype(int))
154             adv_each['female'] = sum(adv['female'].astype(int))
155             adv_each['interested'] = sum(adv['interested'].astype(int))
156             adv_each['neutral'] = sum(adv['neutral'].astype(int))
157             adv_each['notinterested'] = sum(adv['notinterested'].astype(int))
158             mal_adv_detail.append(adv_each)
159         mal_detail['mal_detail'] = mal_adv_detail
160         mal_detail['state_detail'] = state_detail
161         mal_detail['tmpoint_detail'] = tmpoint_detail
162         return (mal_detail)
163 def emoprocessing(data,date,timestamp,url,port,companyname):
164         timestamp_pre,mal_pre_final,sta_pre_final,tm_pre_final =
                getpreviousdata(date,url,port,companyname)
165         error = True
166
167         if len(tm_pre_final) == 0:
168             mal_detail = normalOps(data,date,timestamp)
169             error = False
170             return (mal_detail,error)
171
172         else:
173             data = data.append(tm_pre_final,ignore_index=True)
174             mal_detail = normalOps(data,date,timestamp)
175             error = False
176             return (mal_detail,error)
177
178
179 def main(date,hour,min_,loop=False):
180         source = ace2.read_json('../input.json')
181         url = source['url']
182         port = source['port']
```

```
    path = 'last5min'
    if loop == True:
        path = 'data'
    timestamp = date[0:4]+'/'+date[4:6]+'/'+date[6:]+' '+hour+':'+min_
    print(timestamp)
    timestamp = ace2.datetime.strptime(timestamp, "%Y/%m/%d %H:%M")
    timestamp = timestamp.timestamp()
    data,error= getemodatadaily(date,url,port,path,timestamp)

    premisedata = ace2.getpremises(url,port)

    if error == False:
        data = pd.merge(data, premisedata, on='premisename', how='left')
        companydetails = ace2.pickpremisebycom(url,port)

        for each in companydetails:
            data_each = data[data['premisename'].isin(each['engagepremises
                '])]
            if len(data_each) == 0:
                print('No Data Related to ',each['name'])
            else:

                mal_detail,error = emoprocessing(data_each,date,timestamp,
                    url,port,each['name'])
                # print(mal_detail)
                if error == False:
                    postemo(mal_detail,url,port,each['name'])



    else:
        print('EMO Data Source Got Problem!!')
```

ace2/emo/___init___.py

```
__all__ = ['daily','weekly','monthly',]

import ace2.emo.daily
import ace2.emo.weekly
import ace2.emo.monthly
```

ace2/emo/monthly.py

```
import requests as rq
import json
import ace2
import pandas as pd

def getdailydata(url,port,companyname,month):
```

```
7       request = rq.get ('http://'+url+':'+port+'/ace/api/v2/emotion/
            monthlydata?company='\
8               +companyname+'&month='+month, headers= {'Content-type': '
                    application/json'})
9       # print('http://'+url+':'+port+'/ace/api/v2/emotion/monthlydata?
            company='\
10      #         +companyname+'&month='+month)
11      value = request.text
12      predata = json.loads(value)
13
14      error = True
15      if predata['error'] == False:
16          data = predata['EmotionMonthlylyData']
17
18          if len(data) > 0:
19              tm_pre_final = ace2.pd.DataFrame()
20              for each in data:
21                  data_each = each['data']
22                  date = data_each['date']
23                  tm_predata = ace2.pd.io.json.json_normalize(data_each['
                        tmpoint_detail'])
24                  for each in range(len(tm_predata['adv_detail'])):
25                      tm_each = tm_predata['adv_detail'].iloc[each]
26                      tm_each = ace2.pd.io.json.json_normalize(tm_each)
27                      tm_each['premisename'] = tm_predata['premisename'].
                            iloc[each]
28                      tm_each['state'] = tm_predata['state'].iloc[each]
29                      tm_each['date'] = date
30                      tm_pre_final =tm_pre_final.append(tm_each,ignore_index
                            =True)
31              return(tm_pre_final)
32
33          else:
34              tm_pre_final = ace2.pd.DataFrame()
35              return(tm_pre_final)
36  def normalOps(data,month):
37      year = ace2.datetime.today().strftime("%Y")
38      unique_key = year+str(month)
39
40      # TMpoint  Operations
41      # Group by TMpoint Name
42      tmp_group = data.groupby(by=data.premisename,as_index=False)
43      tmpoint_detail = list()
44      for key,value in tmp_group:
45          group = pd.DataFrame(value)
46          tmp = dict()
47          tmp['premisename'] = group.iloc[0]['premisename']
48          tmp['state'] = group.iloc[0]['state']
49
50          # Group by advert_title
51          adv_group = group.groupby(by=group.advert_title,as_index=False)
52          adv_detail = list()
53          for key,each in adv_group:
54              adv = pd.DataFrame(each)
```

```
            adv_each = dict()
            adv_each['advert_title'] = adv.iloc[0]['advert_title']
            adv_each['male'] = sum(adv['male'].astype(int))
            adv_each['female'] = sum(adv['female'].astype(int))
            adv_each['interested'] = sum(adv['interested'].astype(int))
            adv_each['neutral'] = sum(adv['neutral'].astype(int))
            adv_each['notinterested'] = sum(adv['notinterested'].astype(
                int))
            adv_detail.append(adv_each)

        tmp['adv_detail'] = adv_detail
        tmpoint_detail.append(tmp)
    # print(tmpoint_detail)

    # State Operations
    sta_group = data.groupby(by=data.state,as_index=False)
    state_detail = list()
    for key,value in sta_group:
        group = pd.DataFrame(value)
        sta = dict()
        sta['sta_name'] = group.iloc[0]['state']
        # Group by advert_title
        adv_group = group.groupby(by=group.advert_title,as_index=False)
        adv_detail = list()
        for key,each in adv_group:
            adv = pd.DataFrame(each)
            adv_each = dict()
            adv_each['advert_title'] = adv.iloc[0]['advert_title']
            adv_each['male'] = sum(adv['male'].astype(int))
            adv_each['female'] = sum(adv['female'].astype(int))
            adv_each['interested'] = sum(adv['interested'].astype(int))
            adv_each['neutral'] = sum(adv['neutral'].astype(int))
            adv_each['notinterested'] = sum(adv['notinterested'].astype(
                int))
            adv_detail.append(adv_each)

        sta['adv_detail'] = adv_detail
        state_detail.append(sta)
    # print(state_detail)

    # Malaysia Operations
    mal_detail = dict()
    mal_detail['date'] = unique_key
    mal_group = data.groupby(by=data.advert_title,as_index=False)
    mal_adv_detail = list()
    for key,each in mal_group:
        adv = pd.DataFrame(each)
        adv_each = dict()
        adv_each['advert_title'] = adv.iloc[0]['advert_title']
        adv_each['male'] = sum(adv['male'].astype(int))
        adv_each['female'] = sum(adv['female'].astype(int))
        adv_each['interested'] = sum(adv['interested'].astype(int))
        adv_each['neutral'] = sum(adv['neutral'].astype(int))
        adv_each['notinterested'] = sum(adv['notinterested'].astype(int))
```

```
107             mal_adv_detail.append(adv_each)
108         mal_detail['mal_detail'] = mal_adv_detail
109         mal_detail['state_detail'] = state_detail
110         mal_detail['tmpoint_detail'] = tmpoint_detail
111         return (mal_detail,unique_key)
112 def postemo(mal_detail,url,port,company):
113         url = 'http://'+url+':'+port+'/ace/api/v2/emotion/monthly?company='+
                company
114         headers = {'Content-type': 'application/json'}
115         r = rq.post(url, data=json.dumps(mal_detail), headers=headers)
116         return(r.text)
117
118
119 def main(month):
120         source = ace2.read_json('../input.json')
121         url = source['url']
122         port = source['port']
123
124         companydetails = ace2.pickpremisebycom(url,port)
125         for each in companydetails:
126             tm_pre_final = getdailydata(url,port,each['name'],month)
127             if len(tm_pre_final) == 0:
128                 year = ace2.datetime.today().strftime("%Y")
129                 unique_key = year+'-'+str(month)
130                 print(unique_key,each['name'],'No Weekly Data for week no.',
                        month)
131             else:
132                 mal_detail,unique_key = normalOps(tm_pre_final,month)
133                 text_result = postemo(mal_detail,url,port,each['name'])
134                 print(unique_key,each['name'],text_result)
```

ace2/emo/weekly.py

```
1  import requests as rq
2  import json
3  import ace2
4  import pandas as pd
5
6  def getweeklydata(url,port,companyname,weekno):
7      request = rq.get ('http://'+url+':'+port+'/ace/api/v2/emotion/
           previousweekdata?company='\
8                 +companyname+'&week='+weekno, headers= {'Content-type': '
                      application/json'})
9
10     value = request.text
11     predata = json.loads(value)
12
13     error = True
14     if predata['error'] == False:
15         data = predata['EmotionPreviousWeekData']
16
17         if len(data) > 0:
```

```
18              tm_pre_final = ace2.pd.DataFrame()
19              for each in data:
20                  data_each = each['data']
21                  date = data_each['date']
22                  tm_predata = ace2.pd.io.json.json_normalize(data_each['
                        tmpoint_detail'])
23                  for each in range(len(tm_predata['adv_detail'])):
24                      tm_each = tm_predata['adv_detail'].iloc[each]
25                      tm_each = ace2.pd.io.json.json_normalize(tm_each)
26                      tm_each['premisename'] = tm_predata['premisename'].
                            iloc[each]
27                      tm_each['state'] = tm_predata['state'].iloc[each]
28                      tm_each['date'] = date
29                      tm_pre_final =tm_pre_final.append(tm_each,ignore_index
                            =True)
30              return(tm_pre_final)
31
32          else:
33              tm_pre_final = ace2.pd.DataFrame()
34              return(tm_pre_final)
35  def normalOps(data,weekno):
36      year = ace2.datetime.today().strftime("%Y")
37      unique_key = year+'-'+str(weekno)
38
39      # TMpoint  Operations
40      # Group by TMpoint Name
41      tmp_group = data.groupby(by=data.premisename,as_index=False)
42      tmpoint_detail = list()
43      for key,value in tmp_group:
44          group = pd.DataFrame(value)
45          tmp = dict()
46          tmp['premisename'] = group.iloc[0]['premisename']
47          tmp['state'] = group.iloc[0]['state']
48
49          # Group by advert_title
50          adv_group = group.groupby(by=group.advert_title,as_index=False)
51          adv_detail = list()
52          for key,each in adv_group:
53              adv = pd.DataFrame(each)
54              adv_each = dict()
55              adv_each['advert_title'] = adv.iloc[0]['advert_title']
56              adv_each['male'] = sum(adv['male'].astype(int))
57              adv_each['female'] = sum(adv['female'].astype(int))
58              adv_each['interested'] = sum(adv['interested'].astype(int))
59              adv_each['neutral'] = sum(adv['neutral'].astype(int))
60              adv_each['notinterested'] = sum(adv['notinterested'].astype(
                    int))
61              adv_detail.append(adv_each)
62
63          tmp['adv_detail'] = adv_detail
64          tmpoint_detail.append(tmp)
65      # print(tmpoint_detail)
66
67      # State Operations
```

```
68      sta_group = data.groupby(by=data.state,as_index=False)
69      state_detail = list()
70      for key,value in sta_group:
71          group = pd.DataFrame(value)
72          sta = dict()
73          sta['sta_name'] = group.iloc[0]['state']
74          # Group by advert_title
75          adv_group = group.groupby(by=group.advert_title,as_index=False)
76          adv_detail = list()
77          for key,each in adv_group:
78              adv = pd.DataFrame(each)
79              adv_each = dict()
80              adv_each['advert_title'] = adv.iloc[0]['advert_title']
81              adv_each['male'] = sum(adv['male'].astype(int))
82              adv_each['female'] = sum(adv['female'].astype(int))
83              adv_each['interested'] = sum(adv['interested'].astype(int))
84              adv_each['neutral'] = sum(adv['neutral'].astype(int))
85              adv_each['notinterested'] = sum(adv['notinterested'].astype(
                    int))
86              adv_detail.append(adv_each)
87
88          sta['adv_detail'] = adv_detail
89          state_detail.append(sta)
90      # print(state_detail)
91
92      # Malaysia Operations
93      mal_detail = dict()
94      mal_detail['date'] = unique_key
95      mal_group = data.groupby(by=data.advert_title,as_index=False)
96      mal_adv_detail = list()
97      for key,each in mal_group:
98          adv = pd.DataFrame(each)
99          adv_each = dict()
100         adv_each['advert_title'] = adv.iloc[0]['advert_title']
101         adv_each['male'] = sum(adv['male'].astype(int))
102         adv_each['female'] = sum(adv['female'].astype(int))
103         adv_each['interested'] = sum(adv['interested'].astype(int))
104         adv_each['neutral'] = sum(adv['neutral'].astype(int))
105         adv_each['notinterested'] = sum(adv['notinterested'].astype(int))
106         mal_adv_detail.append(adv_each)
107     mal_detail['mal_detail'] = mal_adv_detail
108     mal_detail['state_detail'] = state_detail
109     mal_detail['tmpoint_detail'] = tmpoint_detail
110     return (mal_detail,unique_key)
111 def postemo(mal_detail,url,port,company):
112     url = 'http://'+url+':'+port+'/ace/api/v2/emotion/weekly?company='+
            company
113     headers = {'Content-type': 'application/json'}
114     r = rq.post(url, data=json.dumps(mal_detail), headers=headers)
115     return(r.text)
116
117
118 def main(weekno):
119     source = ace2.read_json('../input.json')
```

```
    url = source['url']
    port = source['port']

    companydetails = ace2.pickpremisebycom(url,port)
    for each in companydetails:
        tm_pre_final = getweeklydata(url,port,each['name'],weekno)
        if len(tm_pre_final) == 0:
            year = ace2.datetime.today().strftime("%Y")
            unique_key = year+'-'+str(weekno)
            print(unique_key,each['name'],'No Weekly Data for week no.',
                weekno)
        else:
            mal_detail,unique_key = normalOps(tm_pre_final,weekno)
            text_result = postemo(mal_detail,url,port,each['name'])
            print(unique_key,each['name'],text_result)
```

ace2/qms/daily.py

```
import ace2
import os.path
import pandas as pd
import json
import functools as ft
import requests as rq
import multiprocessing as mp


# QMS
def getqmsdata(url, port):
    request = ace2.rq.get('http://'+url+':'+port+'/ace/api/v1/qms/
        last10min',
                          headers={'Content-type': 'application/json'})
    value = request.text
    qmsdata = ace2.json.loads(value)

    if qmsdata['error'] == False:
        qmsdata = qmsdata['QMSLast10Min_data']
        if len(qmsdata) == 0:
            df = pd.DataFrame()
            return df
        else:
            columns_flag = False
            for each in qmsdata:
                premise = each['premise_name']
                data1 = each['data1']
                data2 = each['data2']
                if data1 is None and data2 is None:
                    continue
                data1 = ace2.pd.io.json.json_normalize(data1)
                data2 = ace2.pd.io.json.json_normalize(data2)
                data = data2.append(data1)

```

```
34                  data['premisename'] = premise
35                  data['time_stamp'] = data['time_stamp'].apply(lambda x:
                        ace2.normalizeTimeStamp(x))
36                  data['serveAvg'] = data['serveAvg'].apply(lambda x: ace2.
                        convertTime(x))
37                  data['waitingAvg'] = data['waitingAvg'].apply(lambda x:
                        ace2.convertTime(x))
38                  data['waitingCount'] = data['waitingCount'].astype(float)
39                  data['prevWaiting'] = data['waitingCount'].shift(1)
40                  data['serveCount'] = data['serveCount'].astype(float)
41                  data['prevServe'] = data['serveCount'].shift(1)
42
43                  data['currentVisitor'] = data['waitingCount'].sub(data['
                        prevServe'], fill_value=0)
44                  data['currentVisitor'] = data['currentVisitor'].apply(
                        lambda x: ace2.checkNeg(x))
45                  # data['currentServe'] = data['serveCount'].sub(data['
                        prevServe'], fill_value = 0)
46
47                  data = data.tail(1)
48                  if columns_flag == False:
49                      final_data = data
50                      columns_flag = True
51                  else:
52                      final_data = final_data.append(data, ignore_index=True
                            )
53              final_data.fillna(0, inplace=True)
54
55              return final_data
56
57
58  def getqmsdatafromday(date, hour, min_, url, port):
59      request = rq.get('http://'+url+':'+port+'/ace/api/v1/qms?date=' +
60                       date, headers={'Content-type': 'application/json'})
61      value = request.text
62      qmsdata = json.loads(value)
63
64      if qmsdata['error'] == False:
65          data = qmsdata['qms']
66          if len(data) == 0:
67              return pd.DataFrame()
68          else:
69              timestamp = date + " " + hour + ":" + min_
70              # print(timestamp)
71              date_timestamp = ace2.datetime.strptime(timestamp, "%Y%m%d %H
                    :%M")
72              str_timestamp = date_timestamp.strftime('%s')
73              qms_df = pd.DataFrame()
74              for each in data:
75                  premise = each['premise']
76                  premise_qms = each['qmsdata']
77                  premise_qms = pd.io.json.json_normalize(premise_qms)
78                  premise_qms['premisename'] = premise
```

```
79                  premise_qms['time_stamp'] = premise_qms['time_stamp'].
                        apply(
80                      lambda x: ace2.normalizeTimeStamp(x)).astype(int)
81                  premise_qms['serveAvg'] = premise_qms['serveAvg'].apply(
82                      lambda x: ace2.convertTime(x))
83                  premise_qms['waitingAvg'] = premise_qms['waitingAvg'].
                        apply(
84                      lambda x: ace2.convertTime(x))
85                  premise_qms['waitingCount'] = premise_qms['waitingCount'].
                        astype(float)
86
87                  data1 = premise_qms.loc[premise_qms['time_stamp'] == int(
                        str_timestamp)]
88                  data2 = premise_qms.loc[premise_qms['time_stamp'] == int(
                        str_timestamp)-600]
89                  data = data2.append(data1)
90
91                  if len(data) == 1:
92                      data['prevWaiting'] = 0
93                      data['prevServe'] = 0
94                      data['serveCount'] = data['serveCount'].astype(float)
95                      data['waitingCount'] = data['waitingCount'].astype(
                            float)
96                      data['currentVisitor'] = data['waitingCount']
97                      # data['currentServe'] = data['serveCount']
98                      data['currentVisitor'] = data['currentVisitor'].apply(
99                          lambda x: ace2.checkValue(x))
100                     # data['currentServe'] = data['currentServe'].apply(
                            lambda x: ace2.checkValue(x))
101                     qms_df = qms_df.append(data, ignore_index=True)
102                     qms_df['currentVisitor'] = qms_df['currentVisitor'].
                            apply(
103                         lambda x: ace2.checkNeg(x))
104                     # qms_df['currentServe'] = qms_df['currentServe'].
                            apply(lambda x: ace2.checkNeg(x))
105
106                 elif len(data) == 2:
107                     data['prevWaiting'] = data['waitingCount'].shift(1)
108                     data['serveCount'] = data['serveCount'].astype(float)
109                     data['waitingCount'] = data['waitingCount'].astype(
                            float)
110                     data['prevServe'] = data['serveCount'].shift(1)
111                     data['currentVisitor'] = data['waitingCount'].sub(
112                         data['prevServe'], fill_value=0)
113                     # data['currentServe'] = data['serveCount'].sub(data['
                            prevServe'], fill_value = 0)
114                     data = data.tail(1)
115                     qms_df = qms_df.append(data, ignore_index=True)
116                     qms_df['currentVisitor'] = qms_df['currentVisitor'].
                            apply(
117                         lambda x: ace2.checkNeg(x))
118                     # qms_df['currentServe'] = qms_df['currentServe'].
                            apply(lambda x: ace2.checkNeg(x))
119
```

```
120                 else:
121                     qms_df = qms_df
122
123             qms_df.fillna(0, inplace=True)
124
125             return qms_df
126
127
128 def getqmsdailylast(date, companyname, url, port):
129     request = rq.get('http://'+url+':'+port+'/ace/api/v1/qms/lastdata?date
        ='+date+'&company='+companyname,
130                      headers={'Content-type': 'application/json'})
131     value = request.text
132     predata = json.loads(value)
133     if predata['error'] == False:
134         predata = predata['QMSLastData']
135         if len(predata) == 0:
136             df_mal = pd.DataFrame()
137             df_state = pd.DataFrame()
138             df_tmpoint = pd.DataFrame()
139             return (df_mal, df_state, df_tmpoint)
140         else:
141             predata = predata[0]
142             predata = predata['data']
143             # predata = [data for data in predata if data['companyname']
                == companyname][0]
144             mal_predata = pd.io.json.json_normalize(predata)
145             mal_predata.drop(['tmpoint_detail', 'state_detail'], axis=1,
                inplace=True)
146             state_predata = predata['state_detail']
147             state_predata = pd.io.json.json_normalize(state_predata)
148             tmpoint_predata = predata['tmpoint_detail']
149             tmpoint_predata = pd.io.json.json_normalize(tmpoint_predata)
150
151             return (mal_predata, state_predata, tmpoint_predata)
152
153
154 def getqmsdailyranklast(date, companyname, url, port):
155     request = ace2.rq.get('http://'+url+':'+port+'/ace/api/v1/qms/
        ranklastdata?date='+date+'&company='+companyname,
156                           headers={'Content-type': 'application/json'})
157     value = request.text
158     predata = json.loads(value)
159     if predata['error'] == False:
160         predata = predata['QMSRankLastData']
161         if len(predata) == 0:
162             df = pd.DataFrame()
163             return df
164         else:
165             predata = predata[0]
166             predata = predata['data']
167             # predata = [data for data in predata if data['companyname']
                == companyname][0]
168             return predata
```

```
169
170
171 def postqmsdaily(data, company, url, port):
172     url = 'http://'+url+':'+port+'/ace/api/v1/qms/daily?company='+company
173     headers = {'Content-type': 'application/json'}
174
175     r = rq.post(url, data=json.dumps(data), headers=headers)
176     print(r.text)
177
178
179 def postqmsdailyrank(data, company, url, port):
180     url = 'http://'+url+':'+port+'/ace/api/v1/qms/dailyrank?company='+
        company
181     headers = {'Content-type': 'application/json'}
182     r = rq.post(url, data=json.dumps(data), headers=headers)
183     print(r.text)
184
185
186 # QMS TT (For True Traffic)
187 def getqmsttdata(url, port):
188     request = ace2.rq.get('http://'+url+':'+port+'/ace/api/v1/qmstt/
        last10min',
189                           headers={'Content-type': 'application/json'})
190     value = request.text
191     qmsdata = ace2.json.loads(value)
192
193     qmsdata = qmsdata['QMSTTLast10Min_data']
194     if len(qmsdata) == 0:
195         df = pd.DataFrame()
196         return df
197     else:
198         columns_flag = False
199         for each in qmsdata:
200             premise = each['premise_name']
201             data1 = each['data1']
202             data2 = each['data2']
203             if data1 is None and data2 is None:
204                 continue
205             data1 = ace2.pd.io.json.json_normalize(data1)
206             data2 = ace2.pd.io.json.json_normalize(data2)
207             data = data2.append(data1)
208
209             data['premisename'] = premise
210             data['time_stamp'] = data['time_stamp'].apply(lambda x: ace2.
                normalizeTimeStamp(x))
211             data['serveAvg'] = data['serveAvg'].apply(lambda x: ace2.
                convertTime(x))
212             data['waitingAvg'] = data['waitingAvg'].apply(lambda x: ace2.
                convertTime(x))
213             data['waitingCount'] = data['waitingCount'].astype(float)
214             data['prevWaiting'] = data['waitingCount'].shift(1)
215             data['serveCount'] = data['serveCount'].astype(float)
216             data['prevServe'] = data['serveCount'].shift(1)
```

```
217             data['currentVisitor'] = data['waitingCount'].sub(data['
                    prevServe'], fill_value=0)
218             data['currentVisitor'] = data['currentVisitor'].apply(lambda x
                    : ace2.checkNeg(x))
219             # data['currentServe'] = data['currentServe'].apply(lambda x:
                    ace2.checkNeg(x))
220
221             data = data.tail(1)
222             if columns_flag == False:
223                 final_data = data
224                 columns_flag = True
225             else:
226                 final_data = final_data.append(data, ignore_index=True)
227         final_data.fillna(0, inplace=True)
228         # print(final_data)
229         # import pdb; pdb.set_trace()
230         return final_data
231
232
233 def getqmsttdatafromday(date, hour, min_, url, port):
234     request = rq.get('http://'+url+':'+port+'/ace/api/v1/qmstt/dailydata?
            date='+date,
235                      headers={'Content-type': 'application/json'})
236     value = request.text
237     qmsdata = json.loads(value)
238     data = qmsdata['QMSTTDaily_data']
239     if len(data) == 0:
240         return pd.DataFrame()
241     else:
242         timestamp = date + " " + hour + ":" + min_
243         # print(timestamp)
244         date_timestamp = ace2.datetime.strptime(timestamp, "%Y%m%d %H:%M")
245         str_timestamp = date_timestamp.strftime('%s')
246         qms_df = pd.DataFrame()
247         for each in data:
248             premise = each['premise_name']
249             premise_qms = each['data']
250             premise_qms = pd.io.json.json_normalize(premise_qms)
251             premise_qms['premisename'] = premise
252             premise_qms['time_stamp'] = premise_qms['time_stamp'].apply(
253                 lambda x: ace2.normalizeTimeStamp(x)).astype(int)
254             premise_qms['serveAvg'] = premise_qms['serveAvg'].apply(lambda
                    x: ace2.convertTime(x))
255             premise_qms['waitingAvg'] = premise_qms['waitingAvg'].apply(
256                 lambda x: ace2.convertTime(x))
257             premise_qms['waitingCount'] = premise_qms['waitingCount'].
                    astype(float)
258
259             data1 = premise_qms.loc[premise_qms['time_stamp'] == int(
                    str_timestamp)]
260             data2 = premise_qms.loc[premise_qms['time_stamp'] == int(
                    str_timestamp)-600]
261             data = data2.append(data1)
262             # import pdb; pdb.set_trace()
```

```
263             if len(data) == 1:
264                 data['prevWaiting'] = 0
265                 data['prevServe'] = 0
266                 data['serveCount'] = data['serveCount'].astype(float)
267                 data['waitingCount'] = data['waitingCount'].astype(float)
268                 data['currentVisitor'] = data['waitingCount']
269                 # data['currentServe'] = data['serveCount']
270                 data['currentVisitor'] = data['currentVisitor'].apply(
                        lambda x: ace2.checkValue(x))
271                 # data['currentServe'] = data['currentServe'].apply(lambda
                        x: ace2.checkValue(x))
272                 qms_df = qms_df.append(data, ignore_index=True)
273                 qms_df['currentVisitor'] = qms_df['currentVisitor'].apply(
274                     lambda x: ace2.checkNeg(x))
275                 # qms_df['currentServe'] = qms_df['currentServe'].apply(
                        lambda x: ace2.checkNeg(x))
276
277             elif len(data) == 2:
278                 data['prevWaiting'] = data['waitingCount'].shift(1)
279                 data['serveCount'] = data['serveCount'].astype(float)
280                 data['waitingCount'] = data['waitingCount'].astype(float)
281                 data['prevServe'] = data['serveCount'].shift(1)
282                 data['currentVisitor'] = data['waitingCount'].sub(data['
                        prevServe'], fill_value=0)
283                 # data['currentServe'] = data['serveCount'].sub(data['
                        prevServe'], fill_value = 0)
284                 data = data.tail(1)
285                 qms_df = qms_df.append(data, ignore_index=True)
286                 qms_df['currentVisitor'] = qms_df['currentVisitor'].apply(
287                     lambda x: ace2.checkNeg(x))
288                 # qms_df['currentServe'] = qms_df['currentServe'].apply(
                        lambda x: ace2.checkNeg(x))
289
290             else:
291                 qms_df = qms_df
292
293         qms_df.fillna(0, inplace=True)
294
295         return qms_df
296
297
298 def getqmsttdailylast(date, companyname, url, port):
299     request = rq.get('http://'+url+':'+port+'/ace/api/v1/qmstt/lastdata?
            date='+date+'&company='+companyname,
300                      headers={'Content-type': 'application/json'})
301     value = request.text
302     predata = json.loads(value)
303     if predata['error'] == False:
304         predata = predata['QMSttLastData']
305         if len(predata) == 0:
306             df_mal = pd.DataFrame()
307             df_state = pd.DataFrame()
308             df_tmpoint = pd.DataFrame()
309             return (df_mal, df_state, df_tmpoint)
```

```
310         else:
311             predata = predata[0]
312             predata = predata['data']
313             # predata = [data for data in predata if data['companyname']
                    == companyname][0]
314             mal_predata = pd.io.json.json_normalize(predata)
315             mal_predata.drop(['tmpoint_detail', 'state_detail'], axis=1,
                    inplace=True)
316             state_predata = predata['state_detail']
317             state_predata = pd.io.json.json_normalize(state_predata)
318             tmpoint_predata = predata['tmpoint_detail']
319             tmpoint_predata = pd.io.json.json_normalize(tmpoint_predata)
320
321             return (mal_predata, state_predata, tmpoint_predata)
322
323
324 def postqmsttdaily(data, company, url, port):
325     url = 'http://'+url+':'+port+'/ace/api/v1/qmstt/daily?company='+
            company
326     headers = {'Content-type': 'application/json'}
327
328     r = rq.post(url, data=json.dumps(data), headers=headers)
329     print(r.text)
330
331 # Functional Operations
332
333
334 def buttonsOps(buttons):
335     # print(buttons)
336     a_list = list()
337     for i in range(len(buttons)):
338         a = list(buttons.iloc[i])
339         a_list.append(a)
340
341     a_df = pd.DataFrame(a_list)
342
343     final_sum = list()
344     for i in range(len(a_df.columns)):
345         a_col_list = list(a_df.ix[:, i])
346         new_dictionary = {}
347         for dictionary in a_col_list:
348             if dictionary is not None:
349                 for key, value in dictionary.items():
350                     if key in new_dictionary.keys():
351                         if key == 'value':
352                             new_dictionary[key] = int(value) + int(
                                    new_dictionary[key])
353                         else:
354                             new_dictionary[key] = new_dictionary[key]
355                     else:
356                         new_dictionary[key] = value
357
358         final_sum.append(new_dictionary)
359
```

```
360     return final_sum
361
362
363 def counterSeverAvgOps(counterServeAvg):
364     a_list = list()
365     for i in range(len(counterServeAvg)):
366         a = list(counterServeAvg.iloc[i])
367         a_list.append(a)
368
369     a_df = ace2.pd.DataFrame(a_list)
370
371     final_sum = list()
372     for i in range(len(a_df.columns)):
373         a_col_list = list(a_df.ix[:, i])
374         new_dictionary = {}
375         for dictionary in a_col_list:
376             for key, value in dictionary.items():
377                 if key in new_dictionary.keys():
378                     new_dictionary[key] = ace2.Mintoseconds(value) +
                            new_dictionary[key]
379                 else:
380                     new_dictionary[key] = ace2.Mintoseconds(value)
381
382         for key, value in new_dictionary.items():
383             new_dictionary[key] = ace2.secondstoMin(value/len(a_df))
384
385         final_sum.append(new_dictionary)
386     return final_sum
387
388
389 def qmsoperation(date, qmsdata, premisedata, companyname, url, port, TT):
390
391     if TT == True:
392         mal_predata, state_predata, tmpoint_predata = getqmsttdailylast(
393             date, companyname, url, port)
394     else:
395         mal_predata, state_predata, tmpoint_predata = getqmsdailylast(date
                , companyname, url, port)
396     data = pd.merge(qmsdata, premisedata, on='premisename', how='left')
397
398     timestamp = data.iloc[0]['time_stamp']
399     tmpoints = data.drop(['time_stamp'], axis=1)
400     data['activeCounter'] = data['activeCounter'].astype(int)
401     data['totalCounter'] = data['totalCounter'].astype(int)
402     # No previous data case
403     if len(mal_predata) == 0 and len(state_predata) == 0 and len(
            tmpoint_predata) == 0:
404
405         # Daily General Data
406         # TMpoints Operations
407
408         # tmpoints['count_for_avg'] = tmpoints['serveAvg'].apply(lambda x:
                1)
409         tmpoints['tol_visitor'] = tmpoints['waitingCount']
```

```
410         tmpoints['tol_serve'] = tmpoints['serveCount']
411         tmpoints['totalCounter'] = tmpoints['totalCounter'].astype(int)
412         tmpoints['activeCounter'] = tmpoints['activeCounter'].astype(int)
413         # tmpoints['serveAvg'] = (tmpoints['serveAvg']/tmpoints['
                count_for_avg']).apply(lambda x: ace2.secondstoMin(x))
414         tmpoints['serveAvg'] = tmpoints['serveAvg'].apply(lambda x: ace2.
                secondstoMin(x))
415         # tmpoints['waitingAvg'] = (tmpoints['waitingAvg']/tmpoints['
                count_for_avg']).apply(lambda x: ace2.secondstoMin(x))
416         tmpoints['waitingAvg'] = tmpoints['waitingAvg'].apply(lambda x:
                ace2.secondstoMin(x))
417         tmpoints.drop(['serveCount', 'waitingCount', 'prevWaiting',
418                        'prevServe'], axis=1, inplace=True)
419
420         tmpoint_detail = tmpoints.to_json(orient='records')
421         tmpoint_detail = json.loads(tmpoint_detail)
422
423         # State Operations
424         state_group = data.groupby(by=data.state, as_index=False)
425         state_detail = list()
426         for key, value in state_group:
427             group = pd.DataFrame(value)
428             state = dict()
429
430             state["sta_name"] = group.iloc[0]['state']
431             state["sta_cur_visitor"] = sum(group['currentVisitor'])
432             # state["sta_cur_ser"] = sum(group['currentServe'])
433             state["sta_tol_visitor"] = sum(group['waitingCount'])
434             state["sta_tol_ser"] = sum(group["serveCount"])
435             state["sta_tol_counter"] = sum(group["totalCounter"])
436             state["sta_act_counter"] = sum(group["activeCounter"])
437             state["sta_avg_ast"] = ace2.secondstoMin(sum(group["serveAvg"
                    ])/len(group["serveAvg"]))
438             state["sta_avg_awt"] = ace2.secondstoMin(
439                  sum(group["waitingAvg"])/len(group["waitingAvg"]))
440             state["sta_buttons"] = buttonsOps(group['buttons'])
441             # state["sta_counter_avg_ast"] = counterSeverAvgOps(group['
                    counterServeAvg']) # Dontneed yet
442             sta_counter_details = group[['premisename', 'totalCounter', '
                    activeCounter']]
443             sta_counter_details = sta_counter_details.to_json(orient='
                    records')
444             sta_counter_details = json.loads(sta_counter_details)
445             state['sta_counter_details'] = sta_counter_details
446
447             state_detail.append(state)
448
449         state_detail_json = json.dumps(state_detail)
450         state_detail_json = json.loads(state_detail_json)
451
452         # Malaysian Operations
453         state_df = pd.DataFrame.from_dict(state_detail)
454         state_df['sta_avg_ast'] = state_df['sta_avg_ast'].apply(lambda x:
                ace2.Mintoseconds(x))
```

```
455         state_df['sta_avg_awt'] = state_df['sta_avg_awt'].apply(lambda x:
                ace2.Mintoseconds(x))
456
457         data_obj = dict()
458         data_obj['date'] = ace2.datetime.fromtimestamp(int(timestamp)).
                strftime('%Y%m%d')
459         data_obj['timestamp'] = str(timestamp)
460         data_obj['mal_cur_visitor'] = sum(state_df['sta_cur_visitor'])
461         # data_obj['mal_cur_ser'] = sum(state_df['sta_cur_ser'])
462         data_obj['mal_tol_visitor'] = sum(state_df['sta_tol_visitor'])
463         data_obj['mal_tol_ser'] = sum(state_df['sta_tol_ser'])
464         data_obj['mal_avg_ast'] = ace2.secondstoMin(
465             sum(state_df['sta_avg_ast'])/len(state_df['sta_avg_ast']))
466         data_obj['mal_avg_awt'] = ace2.secondstoMin(
467             sum(state_df['sta_avg_awt'])/len(state_df['sta_avg_awt']))
468         data_obj["mal_tol_counter"] = sum(state_df["sta_tol_counter"])
469         data_obj["mal_act_counter"] = sum(state_df["sta_act_counter"])
470         data_obj["mal_buttons"] = buttonsOps(state_df['sta_buttons'])
471         # data_obj["mal_counter_avg_ast"] = counterSeverAvgOps(state_df['
                sta_counter_avg_ast']) # Dontneed yet
472         data_obj['state_detail'] = state_detail_json
473         data_obj['tmpoint_detail'] = tmpoint_detail
474
475         mal_rank = calculaterank(tmpoints, date, timestamp)
476
477         return (data_obj, mal_rank)
478     # Previous data case
479     elif len(mal_predata) > 0 and len(state_predata) > 0 and len(
            tmpoint_predata) > 0:
480         timestamp_check = int(mal_predata.iloc[0]['timestamp'])
481         diff_time = timestamp - timestamp_check
482         print("Time Diff: ", diff_time)
483         print(timestamp, timestamp_check)
484         if diff_time == 600:
485             # General Data Processing
486             # TMpoints Operations
487             # tmpoints['count_for_avg'] = 1 + tmpoint_predata['
                    count_for_avg']
488             tmpoints['tol_visitor'] = tmpoints['waitingCount']
489             tmpoints['tol_serve'] = tmpoints['serveCount']
490             tmpoints = tmpoints.fillna(method='ffill')
491             tmpoints['totalCounter'] = tmpoints['totalCounter'].astype(int
                    )
492             tmpoints['activeCounter'] = tmpoints['activeCounter'].astype(
                    int)
493             # tmpoints['serveAvg'] = (tmpoints['serveAvg']/tmpoints['
                    count_for_avg']).apply(lambda x: ace2.secondstoMin(x))
494             tmpoints['serveAvg'] = tmpoints['serveAvg'].apply(lambda x:
                    ace2.secondstoMin(x))
495             # tmpoints['waitingAvg'] = (tmpoints['waitingAvg']/tmpoints['
                    count_for_avg']).apply(lambda x: ace2.secondstoMin(x))
496             tmpoints['waitingAvg'] = tmpoints['waitingAvg'].apply(lambda x
                    : ace2.secondstoMin(x))
497             tmpoints.drop(['serveCount', 'waitingCount', 'prevWaiting',
```

```
                    'prevServe'], axis=1, inplace=True)
            tmpoint_detail = tmpoints.to_json(orient='records')
            tmpoint_detail = json.loads(tmpoint_detail)

            # State Operations
            state_group = data.groupby(by=data.state, as_index=False)
            state_detail = list()
            for key, value in state_group:
                group = pd.DataFrame(value)
                state = dict()
                state["sta_name"] = group.iloc[0]['state']
                state["sta_cur_visitor"] = sum(group['currentVisitor'])
                # state["sta_cur_ser"] = sum(group['currentServe'])
                state["sta_buttons"] = buttonsOps(group['buttons'])
                state["sta_tol_counter"] = sum(group["totalCounter"])
                state["sta_act_counter"] = sum(group["activeCounter"])
                sta_counter_details = group[['premisename', 'totalCounter'
                    , 'activeCounter']]
                sta_counter_details = sta_counter_details.to_json(orient='
                    records')
                sta_counter_details = json.loads(sta_counter_details)
                state['sta_counter_details'] = sta_counter_details
                state["sta_tol_visitor"] = sum(group["waitingCount"])
                state["sta_tol_ser"] = sum(group['serveCount'])
                state["sta_avg_ast"] = ace2.secondstoMin(
                    sum(group["serveAvg"])/len(group["serveAvg"]))
                state["sta_avg_awt"] = ace2.secondstoMin(
                    sum(group["waitingAvg"])/len(group["waitingAvg"]))

                state_detail.append(state)

            state_detail_json = json.dumps(state_detail)
            state_detail_json = json.loads(state_detail_json)

            # Malaysian Operations
            state_df = pd.DataFrame.from_dict(state_detail)
            state_df['sta_avg_ast'] = state_df['sta_avg_ast'].apply(lambda
                x: ace2.Mintoseconds(x))
            state_df['sta_avg_awt'] = state_df['sta_avg_awt'].apply(lambda
                x: ace2.Mintoseconds(x))

            data_obj = dict()
            data_obj['date'] = date
            data_obj['timestamp'] = str(timestamp)
            data_obj['mal_cur_visitor'] = sum(state_df['sta_cur_visitor'])
            # data_obj['mal_cur_ser'] = sum(state_df['sta_cur_ser'])
            data_obj['mal_tol_visitor'] = sum(state_df['sta_tol_visitor'])
            data_obj['mal_tol_ser'] = sum(state_df['sta_tol_ser'])
            data_obj['mal_avg_ast'] = ace2.secondstoMin(
                sum(state_df['sta_avg_ast'])/len(state_df['sta_avg_ast']))
            data_obj['mal_avg_awt'] = ace2.secondstoMin(
                sum(state_df['sta_avg_awt'])/len(state_df['sta_avg_awt']))
            data_obj["mal_buttons"] = buttonsOps(state_df['sta_buttons'])
            data_obj["mal_tol_counter"] = sum(state_df["sta_tol_counter"])
```

```
548             data_obj["mal_act_counter"] = sum(state_df["sta_act_counter"])
549             data_obj['state_detail'] = state_detail_json
550             data_obj['tmpoint_detail'] = tmpoint_detail
551 
552             mal_rank = calculaterank(tmpoints, date, timestamp)
553 
554             return (data_obj, mal_rank)
555         elif diff_time > 600 or diff_time == 0:
556             # General Info
557             data_obj = dict()
558             state_detail_json = state_predata.to_json(orient='records')
559             state_detail_json = json.loads(state_detail_json)
560 
561             tmpoints_detail_json = tmpoint_predata.to_json(orient='records
                    ')
562             tmpoints_detail_json = json.loads(tmpoints_detail_json)
563 
564             data_obj['date'] = date
565             if diff_time == 0:
566                 data_obj['timestamp'] = str(timestamp_check + 600)
567             else:
568                 data_obj['timestamp'] = str(timestamp_check + diff_time)
569 
570             data_obj['mal_cur_visitor'] = str(mal_predata.iloc[0]['
                    mal_cur_visitor'])
571             # data_obj['mal_cur_ser'] = str(mal_predata.iloc[0]['
                    mal_cur_ser'])
572             data_obj['mal_tol_visitor'] = str(mal_predata.iloc[0]['
                    mal_tol_visitor'])
573             data_obj['mal_tol_ser'] = str(mal_predata.iloc[0]['mal_tol_ser
                    '])
574             data_obj['mal_avg_ast'] = str(mal_predata.iloc[0]['mal_avg_ast
                    '])
575             data_obj['mal_avg_awt'] = str(mal_predata.iloc[0]['mal_avg_awt
                    '])
576             data_obj["mal_buttons"] = list(mal_predata['mal_buttons'])[0]
577             data_obj["mal_tol_counter"] = str(mal_predata.iloc[0]["
                    mal_tol_counter"])
578             data_obj["mal_act_counter"] = str(mal_predata.iloc[0]["
                    mal_act_counter"])
579             data_obj['state_detail'] = state_detail_json
580             data_obj['tmpoint_detail'] = tmpoints_detail_json
581 
582             # Rank Info
583             pre_rank = getqmsdailyranklast(date, companyname, url, port)
                    [0]
584             # print(pre_rank)
585             if diff_time <= 0:
586                 pre_rank['timestamp'] = str(timestamp_check + 600)
587             else:
588                 pre_rank['timestamp'] = str(timestamp_check + diff_time)
589             pre_rank['date'] = date
590             mal_rank = pre_rank
591 
```

```
592             return (data_obj, mal_rank)
593         else:
594             data_obj = dict()
595             mal_rank = dict()
596             print("Unsupported Conditions! Time different might be
                    negative value.")
597             return (data_obj, mal_rank)
598     else:
599         print("Key columns Premise not in both dataframes.")
600
601
602 def qmsoperation5MIN(date, qmsdata, premisedata, companyname, url, port,
        TT):
603
604     if TT == True:
605         mal_predata, state_predata, tmpoint_predata = getqmsttdailylast(
606             date, companyname, url, port)
607     else:
608         mal_predata, state_predata, tmpoint_predata = getqmsdailylast(date
                , companyname, url, port)
609     data = pd.merge(qmsdata, premisedata, on='premisename', how='left')
610
611     timestamp = data.iloc[0]['time_stamp']
612     tmpoints = data.drop(['time_stamp'], axis=1)
613     data['activeCounter'] = data['activeCounter'].astype(int)
614     data['totalCounter'] = data['totalCounter'].astype(int)
615     # No previous data case
616     if len(mal_predata) == 0 and len(state_predata) == 0 and len(
            tmpoint_predata) == 0:
617
618         # Daily General Data
619         # TMpoints Operations
620
621         # tmpoints['count_for_avg'] = tmpoints['serveAvg'].apply(lambda x:
                1)
622         tmpoints['tol_visitor'] = tmpoints['waitingCount']
623         tmpoints['tol_serve'] = tmpoints['serveCount']
624         tmpoints['totalCounter'] = tmpoints['totalCounter'].astype(int)
625         tmpoints['activeCounter'] = tmpoints['activeCounter'].astype(int)
626         # tmpoints['serveAvg'] = (tmpoints['serveAvg']/tmpoints['
                count_for_avg']).apply(lambda x: ace2.secondstoMin(x))
627         tmpoints['serveAvg'] = tmpoints['serveAvg'].apply(lambda x: ace2.
                secondstoMin(x))
628         # tmpoints['waitingAvg'] = (tmpoints['waitingAvg']/tmpoints['
                count_for_avg']).apply(lambda x: ace2.secondstoMin(x))
629         tmpoints['waitingAvg'] = tmpoints['waitingAvg'].apply(lambda x:
                ace2.secondstoMin(x))
630         tmpoints.drop(['serveCount', 'waitingCount', 'prevWaiting',
631                        'prevServe'], axis=1, inplace=True)
632
633         tmpoint_detail = tmpoints.to_json(orient='records')
634         tmpoint_detail = json.loads(tmpoint_detail)
635
636         # State Operations
```

```
637         state_group = data.groupby(by=data.state, as_index=False)
638         state_detail = list()
639         for key, value in state_group:
640             group = pd.DataFrame(value)
641             state = dict()
642 
643             state["sta_name"] = group.iloc[0]['state']
644             state["sta_cur_visitor"] = sum(group['currentVisitor'])
645             # state["sta_cur_ser"] = sum(group['currentServe'])
646             state["sta_tol_visitor"] = sum(group['waitingCount'])
647             state["sta_tol_ser"] = sum(group["serveCount"])
648             state["sta_tol_counter"] = sum(group["totalCounter"])
649             state["sta_act_counter"] = sum(group["activeCounter"])
650             state["sta_avg_ast"] = ace2.secondstoMin(sum(group["serveAvg"
                ])/len(group["serveAvg"]))
651             state["sta_avg_awt"] = ace2.secondstoMin(
652                 sum(group["waitingAvg"])/len(group["waitingAvg"]))
653             state["sta_buttons"] = buttonsOps(group['buttons'])
654             # state["sta_counter_avg_ast"] = counterSeverAvgOps(group['
                counterServeAvg']) # Dontneed yet
655             sta_counter_details = group[['premisename', 'totalCounter', '
                activeCounter']]
656             sta_counter_details = sta_counter_details.to_json(orient='
                records')
657             sta_counter_details = json.loads(sta_counter_details)
658             state['sta_counter_details'] = sta_counter_details
659 
660             state_detail.append(state)
661 
662         state_detail_json = json.dumps(state_detail)
663         state_detail_json = json.loads(state_detail_json)
664 
665         # Malaysian Operations
666         state_df = pd.DataFrame.from_dict(state_detail)
667         state_df['sta_avg_ast'] = state_df['sta_avg_ast'].apply(lambda x:
             ace2.Mintoseconds(x))
668         state_df['sta_avg_awt'] = state_df['sta_avg_awt'].apply(lambda x:
             ace2.Mintoseconds(x))
669 
670         data_obj = dict()
671         data_obj['date'] = ace2.datetime.fromtimestamp(int(timestamp)).
             strftime('%Y%m%d')
672         data_obj['timestamp'] = str(timestamp)
673         data_obj['mal_cur_visitor'] = sum(state_df['sta_cur_visitor'])
674         # data_obj['mal_cur_ser'] = sum(state_df['sta_cur_ser'])
675         data_obj['mal_tol_visitor'] = sum(state_df['sta_tol_visitor'])
676         data_obj['mal_tol_ser'] = sum(state_df['sta_tol_ser'])
677         data_obj['mal_avg_ast'] = ace2.secondstoMin(
678             sum(state_df['sta_avg_ast'])/len(state_df['sta_avg_ast']))
679         data_obj['mal_avg_awt'] = ace2.secondstoMin(
680             sum(state_df['sta_avg_awt'])/len(state_df['sta_avg_awt']))
681         data_obj["mal_tol_counter"] = sum(state_df["sta_tol_counter"])
682         data_obj["mal_act_counter"] = sum(state_df["sta_act_counter"])
683         data_obj["mal_buttons"] = buttonsOps(state_df['sta_buttons'])
```

```
684         # data_obj["mal_counter_avg_ast"] = counterSeverAvgOps(state_df['
                sta_counter_avg_ast']) # Dontneed yet
685         data_obj['state_detail'] = state_detail_json
686         data_obj['tmpoint_detail'] = tmpoint_detail
687 
688         mal_rank = calculaterank(tmpoints, date, timestamp)
689 
690         return (data_obj, mal_rank)
691     # Previous data case
692     elif len(mal_predata) > 0 and len(state_predata) > 0 and len(
            tmpoint_predata) > 0:
693         # timestamp_check = int(mal_predata.iloc[0]['timestamp'])
694         # diff_time = timestamp - timestamp_check
695         # print("Time Diff: ",diff_time)
696         # print(timestamp,timestamp_check)
697 
698         # General Data Processing
699         # TMpoints Operations
700         # tmpoints['count_for_avg'] = 1 + tmpoint_predata['count_for_avg']
701         tmpoints['tol_visitor'] = tmpoints['waitingCount']
702         tmpoints['tol_serve'] = tmpoints['serveCount']
703         tmpoints = tmpoints.fillna(method='ffill')
704         tmpoints['totalCounter'] = tmpoints['totalCounter'].astype(int)
705         tmpoints['activeCounter'] = tmpoints['activeCounter'].astype(int)
706         # tmpoints['serveAvg'] = (tmpoints['serveAvg']/tmpoints['
                count_for_avg']).apply(lambda x: ace2.secondstoMin(x))
707         tmpoints['serveAvg'] = tmpoints['serveAvg'].apply(lambda x: ace2.
                secondstoMin(x))
708         # tmpoints['waitingAvg'] = (tmpoints['waitingAvg']/tmpoints['
                count_for_avg']).apply(lambda x: ace2.secondstoMin(x))
709         tmpoints['waitingAvg'] = tmpoints['waitingAvg'].apply(lambda x:
                ace2.secondstoMin(x))
710         tmpoints.drop(['serveCount', 'waitingCount', 'prevWaiting',
711                        'prevServe'], axis=1, inplace=True)
712         tmpoint_detail = tmpoints.to_json(orient='records')
713         tmpoint_detail = json.loads(tmpoint_detail)
714 
715         # State Operations
716         state_group = data.groupby(by=data.state, as_index=False)
717         state_detail = list()
718         for key, value in state_group:
719             group = pd.DataFrame(value)
720             state = dict()
721             state["sta_name"] = group.iloc[0]['state']
722             state["sta_cur_visitor"] = sum(group['currentVisitor'])
723             # state["sta_cur_ser"] = sum(group['currentServe'])
724             state["sta_buttons"] = buttonsOps(group['buttons'])
725             state["sta_tol_counter"] = sum(group["totalCounter"])
726             state["sta_act_counter"] = sum(group["activeCounter"])
727             sta_counter_details = group[['premisename', 'totalCounter', '
                    activeCounter']]
728             sta_counter_details = sta_counter_details.to_json(orient='
                    records')
729             sta_counter_details = json.loads(sta_counter_details)
```

```
730             state['sta_counter_details'] = sta_counter_details
731             state["sta_tol_visitor"] = sum(group["waitingCount"])
732             state["sta_tol_ser"] = sum(group['serveCount'])
733             state["sta_avg_ast"] = ace2.secondstoMin(sum(group["serveAvg"
                    ])/len(group["serveAvg"]))
734             state["sta_avg_awt"] = ace2.secondstoMin(
735                 sum(group["waitingAvg"])/len(group["waitingAvg"]))
736
737             state_detail.append(state)
738
739         state_detail_json = json.dumps(state_detail)
740         state_detail_json = json.loads(state_detail_json)
741
742         # Malaysian Operations
743         state_df = pd.DataFrame.from_dict(state_detail)
744         state_df['sta_avg_ast'] = state_df['sta_avg_ast'].apply(lambda x:
                ace2.Mintoseconds(x))
745         state_df['sta_avg_awt'] = state_df['sta_avg_awt'].apply(lambda x:
                ace2.Mintoseconds(x))
746
747         data_obj = dict()
748         data_obj['date'] = date
749         data_obj['timestamp'] = str(timestamp)
750         data_obj['mal_cur_visitor'] = sum(state_df['sta_cur_visitor'])
751         # data_obj['mal_cur_ser'] = sum(state_df['sta_cur_ser'])
752         data_obj['mal_tol_visitor'] = sum(state_df['sta_tol_visitor'])
753         data_obj['mal_tol_ser'] = sum(state_df['sta_tol_ser'])
754         data_obj['mal_avg_ast'] = ace2.secondstoMin(
755             sum(state_df['sta_avg_ast'])/len(state_df['sta_avg_ast']))
756         data_obj['mal_avg_awt'] = ace2.secondstoMin(
757             sum(state_df['sta_avg_awt'])/len(state_df['sta_avg_awt']))
758         data_obj["mal_buttons"] = buttonsOps(state_df['sta_buttons'])
759         data_obj["mal_tol_counter"] = sum(state_df["sta_tol_counter"])
760         data_obj["mal_act_counter"] = sum(state_df["sta_act_counter"])
761         data_obj['state_detail'] = state_detail_json
762         data_obj['tmpoint_detail'] = tmpoint_detail
763
764         mal_rank = calculaterank(tmpoints, date, timestamp)
765         return (data_obj, mal_rank)
766
767     else:
768         print("Key columns Premise not in both dataframes.")
769         data_obj = dict()
770         mal_rank = dict()
771         return (data_obj, mal_rank)
772
773
774 def calculaterank(tmpoints, date, timestamp):
775     # Daily Rank Data
776     # State Operations
777     sta_rank_group = tmpoints.groupby(by=tmpoints.state, as_index=False)
778     sta_rank_detail = list()
779     for key, value in sta_rank_group:
780         group = pd.DataFrame(value)
```

```
781             state = dict()
782             state['sta_name'] = group.iloc[0]['state']
783             ser_rank = group.sort_values(by=['tol_serve'], ascending=False)
784             ser_rank.drop(['tol_visitor', 'serveAvg', 'waitingAvg', '
                    currentVisitor', 'state', 'buttons',
785                            'counterServeAvg', 'activeCounter', 'totalCounter'
                               ],
786                           axis=1, inplace=True)
787             ser_rank['tol_serve'] = ser_rank['tol_serve'].astype(str)
788             ser_rank_json = ser_rank.to_json(orient='records')
789             ser_rank_json = json.loads(ser_rank_json)
790             state['sta_ser_rank'] = ser_rank_json
791 
792             cur_rank = group.sort_values(by=['tol_visitor'], ascending=False)
793             cur_rank.drop(['tol_serve', 'serveAvg', 'buttons',
794                            'counterServeAvg', 'waitingAvg', 'currentVisitor',
                               'state', 'activeCounter', 'totalCounter', '
                               buttons',
795                            'counterServeAvg', ],
796                           axis=1, inplace=True)
797             cur_rank['tol_visitor'] = cur_rank['tol_visitor'].astype(str)
798             cur_rank_json = cur_rank.to_json(orient='records')
799             cur_rank_json = json.loads(cur_rank_json)
800             state['sta_vis_rank'] = cur_rank_json
801             sta_rank_detail.append(state)
802 
803         mal_ser_rank = tmpoints.sort_values(by=['tol_serve'], ascending=False)
804         mal_ser_rank.drop(['serveAvg', 'buttons',
805                            'counterServeAvg', 'waitingAvg', 'currentVisitor',
                               'state', 'activeCounter', 'totalCounter'],
806                           axis=1, inplace=True)
807 
808         # Rank data Processing
809         mal_rank = dict()
810         mal_rank['state_rank'] = sta_rank_detail
811         mal_ser_rank['tol_serve'] = mal_ser_rank['tol_serve'].astype(str)
812         mal_ser_rank_json = mal_ser_rank.to_json(orient='records')
813         mal_ser_rank_json = json.loads(mal_ser_rank_json)
814         mal_rank['mal_ser_rank'] = mal_ser_rank_json
815 
816         mal_vis_rank = tmpoints.sort_values(by=['tol_visitor'], ascending=
                False)
817         mal_vis_rank.drop(['tol_serve', 'serveAvg', 'buttons',
818                            'counterServeAvg', 'waitingAvg', 'currentVisitor',
                               'state', 'activeCounter', 'totalCounter'],
819                           axis=1, inplace=True)
820         mal_vis_rank['tol_visitor'] = mal_vis_rank['tol_visitor'].astype(str)
821         mal_vis_rank_json = mal_vis_rank.to_json(orient='records')
822         mal_vis_rank_json = json.loads(mal_vis_rank_json)
823         mal_rank['mal_vis_rank'] = mal_vis_rank_json
824         mal_rank['date'] = date
825         mal_rank['timestamp'] = str(timestamp)
826         return mal_rank
827 
```

```
828
829 def withoutMPQMS(date, companydetails, qmsdata, premisedata, url, port, TT
        , loop):
830     for each in companydetails:
831         data = qmsdata[qmsdata['premisename'].isin(each['qmspremises'])]
832         if len(data) > 0:
833             if loop == True:
834                 data_obj, mal_rank = qmsoperation(
835                     date, data, premisedata, each['name'], url, port, TT)
836             else:
837                 data_obj, mal_rank = qmsoperation5MIN(
838                     date, data, premisedata, each['name'], url, port, TT)
839             if len(data_obj) > 0:
840                 if TT == True:
841                     postqmsttdaily(data_obj, each['name'], url, port)
842                 else:
843                     postqmsdaily(data_obj, each['name'], url, port)
844                     postqmsdailyrank(mal_rank, each['name'], url, port)
845
846
847 def main(date, hour, min_, loop=False, TT=False, TTloop=False):
848     print(date + " " + hour + ":" + min_)
849     source = ace2.read_json('../input.json')
850     url = source['url']
851     port = source['port']
852     if loop == True:
853         qmsdata = getqmsdatafromday(date, hour, min_, url, port)
854
855     else:
856         if TT == True:
857             if TTloop == True:
858                 qmsdata = getqmsttdatafromday(date, hour, min_, url, port)
859             else:
860                 qmsdata = getqmsttdata(url, port)
861         else:
862             date = ace2.datetime.now().strftime("%Y%m%d")
863             qmsdata = getqmsdata(url, port)
864
865     # pdb.set_trace()
866     if len(qmsdata) == 0:
867         print("NO QMS DATA")
868     else:
869         companydetails = ace2.pickpremisebycom(url, port)
870         premisedata = ace2.getpremises(url, port)
871
872         # With multiprocessing
873         withoutMPQMS(date, companydetails, qmsdata, premisedata, url, port
            , TT, loop)
874         # mal_predata,state_predata,tmpoint_predata = getqmsdailylast(date
            ,'TSSSB',url,port))
875         # print(mal_predata)
```

ace2/qms/___init___.py

```

__all__ = ['daily','weekly','monthly']

import ace2.qms.daily
import ace2.qms.weekly
import ace2.qms.monthly
```

ace2/qms/monthly.py

```
import ace2
import os.path
import pandas as pd
import json
import requests as rq

# QMS
def getqmsmonthlydata(companyname,month,url,port):
    request = rq.get ('http://'+url+':'+port+'/ace/api/v1/qms/monthlydata?company='+companyname+'&month='+month, \
         headers= {'Content-type': 'application/json'})
    value = request.text
    data = json.loads(value)
    if data['error'] == False:
        data = data['QMSMonthlyData']
        if len(data) == 0:

            tmp_data = pd.DataFrame()
            sta_data = pd.DataFrame()
            mal_data = pd.DataFrame()
            return (mal_data,sta_data,tmp_data)
        else:
            columns_flag = False
            for each in data:
                dailydata = each['data']
                dailytmpoints = dailydata['tmpoint_detail']
                dailystates = dailydata['state_detail']
                dailystates = pd.io.json.json_normalize(dailystates)
                dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
                dailymal = pd.io.json.json_normalize(dailydata)
                dailymal.drop(['tmpoint_detail','state_detail'], axis = 1, inplace = True)
                if columns_flag == False:
                    tmp_data = dailytmpoints
                    sta_data = dailystates
                    mal_data = dailymal
                    columns_flag= True
                else:

```

```
38                      tmp_data = tmp_data.append(dailytmpoints, ignore_index
                            =True)
39                      sta_data = sta_data.append(dailystates, ignore_index=
                            True)
40                      mal_data = mal_data.append(dailymal, ignore_index=True
                            )
41              return (mal_data,sta_data,tmp_data)
42  def postqmsmonthly(data,companyname,url,port):
43      url = 'http://'+url+':'+port+'/ace/api/v1/qms/monthly?company='+
            companyname
44      headers = {'Content-type': 'application/json'}
45      r = rq.post(url, data=json.dumps(data), headers=headers)
46      print(r.text)
47  def postqmsmonthlyrank(data,companyname,url,port):
48      url = 'http://'+url+':'+port+'/ace/api/v1/qms/monthlyrank?company='+
            companyname
49      headers = {'Content-type': 'application/json'}
50      r = rq.post(url, data=json.dumps(data), headers=headers)
51      print(r.text)
52
53  # QMS TT
54  def getqmsttmonthlydata(companyname,month,url,port):
55      request = rq.get ('http://'+url+':'+port+'/ace/api/v1/qmstt/
            monthlydata?company='+companyname+'&month='+month, \
56              headers= {'Content-type': 'application/json'})
57      value = request.text
58      data = json.loads(value)
59      if data['error'] == False:
60          data = data['QMSttMonthlyData']
61          if len(data) == 0:
62
63              tmp_data = pd.DataFrame()
64              sta_data = pd.DataFrame()
65              mal_data = pd.DataFrame()
66              return (mal_data,sta_data,tmp_data)
67          else:
68              columns_flag = False
69              for each in data:
70                  dailydata = each['data']
71                  dailytmpoints = dailydata['tmpoint_detail']
72                  dailystates = dailydata['state_detail']
73                  dailystates = pd.io.json.json_normalize(dailystates)
74                  dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
75                  dailymal = pd.io.json.json_normalize(dailydata)
76                  dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                        inplace = True)
77                  if columns_flag == False:
78                      tmp_data = dailytmpoints
79                      sta_data = dailystates
80                      mal_data = dailymal
81                      columns_flag= True
82                  else:
83
```

```
84                      tmp_data = tmp_data.append(dailytmpoints, ignore_index
                            =True)
85                      sta_data = sta_data.append(dailystates, ignore_index=
                            True)
86                      mal_data = mal_data.append(dailymal, ignore_index=True
                            )
87              return (mal_data,sta_data,tmp_data)
88  def postqmsttmonthly(data,companyname,url,port):
89      url = 'http://'+url+':'+port+'/ace/api/v1/qmstt/monthly?company='+
            companyname
90      headers = {'Content-type': 'application/json'}
91      r = rq.post(url, data=json.dumps(data), headers=headers)
92      print(r.text)
93
94  # Functional Operations
95  def buttonsOps(buttons) :
96      # print(buttons)
97      a_list = list()
98      for i in range(len(buttons)):
99          a = list(buttons.iloc[i])
100         a_list.append(a)
101
102     a_df = pd.DataFrame(a_list)
103
104     final_sum = list()
105     for i in range(len(a_df.columns)):
106         a_col_list = list(a_df.ix[:,i])
107         new_dictionary = {}
108         for dictionary in a_col_list:
109             if dictionary is not None:
110                 for key, value in dictionary.items():
111                     if key in new_dictionary.keys():
112                         if key == 'value':
113                             new_dictionary[key] = int(value) + int(
                                    new_dictionary[key])
114                         else:
115                             new_dictionary[key] = new_dictionary[key]
116                     else:
117                         new_dictionary[key] = value
118
119
120
121         final_sum.append(new_dictionary)
122
123     return final_sum
124 def counterSeverAvgOps(counterServeAvg) :
125     a_list = list()
126     for i in range(len(counterServeAvg)):
127         a = list(counterServeAvg.iloc[i])
128         a_list.append(a)
129
130     a_df = ace2.pd.DataFrame(a_list)
131
132     final_sum = list()
```

```
133     for i in range(len(a_df.columns)):
134         a_col_list = list(a_df.ix[:,i])
135         new_dictionary = {}
136         for dictionary in a_col_list:
137             for key, value in dictionary.items():
138                 if key in new_dictionary.keys():
139                     if key == 'value':
140                         new_dictionary[key] =   ace2.Mintoseconds(value) +
                                new_dictionary[key]
141                     if key == 'name':
142                         new_dictionary[key] = value
143                 else:
144                     if key == 'value':
145                         new_dictionary[key] = ace2.Mintoseconds(value)
146                     if key == 'name':
147                         new_dictionary[key] = value
148         for key, value in new_dictionary.items():
149             if key == 'value':
150                 new_dictionary[key] = ace2.secondstoMin(value/len(a_df))
151
152
153         final_sum.append(new_dictionary)
154     return final_sum
155 def calculaterank(tmpoints,date):
156     # Daily Rank Data
157     # State Operations
158     sta_rank_group = tmpoints.groupby(by=tmpoints.state,as_index=False)
159     sta_rank_detail = list()
160     for key,value in sta_rank_group:
161         group = pd.DataFrame(value)
162         state = dict()
163         state['sta_name'] = group.iloc[0]['state']
164         ser_rank = group.sort_values(by=['tol_serve'], ascending=False)
165         ser_rank.drop(['tol_visitor','buttons','serveAvg','waitingAvg','
              state'],\
166                       axis = 1, inplace = True)
167         ser_rank['tol_serve'] = ser_rank['tol_serve'].astype(str)
168         ser_rank_json = ser_rank.to_json(orient='records')
169         ser_rank_json = json.loads(ser_rank_json)
170         state['sta_ser_rank'] = ser_rank_json
171
172
173         cur_rank = group.sort_values(by=['tol_visitor'], ascending=False)
174         cur_rank.drop(['tol_serve','buttons','serveAvg','waitingAvg','
              state'],\
175                       axis = 1, inplace = True)
176         cur_rank['tol_visitor'] = cur_rank['tol_visitor'].astype(str)
177         cur_rank_json = cur_rank.to_json(orient='records')
178         cur_rank_json = json.loads(cur_rank_json)
179         state['sta_vis_rank'] = cur_rank_json
180         sta_rank_detail.append(state)
181
182     mal_ser_rank = tmpoints.sort_values(by=['tol_serve'], ascending=False)
```

```
183     mal_ser_rank.drop(['tol_visitor','buttons','serveAvg','waitingAvg','
            state'],\
184                 axis = 1, inplace = True)
185 
186     # Rank data Processing
187     mal_rank = dict()
188     mal_rank['state_rank'] =  sta_rank_detail
189     mal_ser_rank['tol_serve'] = mal_ser_rank['tol_serve'].astype(str)
190     mal_ser_rank_json = mal_ser_rank.to_json(orient='records')
191     mal_ser_rank_json = json.loads(mal_ser_rank_json)
192     mal_rank['mal_ser_rank'] = mal_ser_rank_json
193 
194     mal_vis_rank = tmpoints.sort_values(by=['tol_visitor'], ascending=
            False)
195     mal_vis_rank.drop(['tol_serve','buttons','serveAvg','waitingAvg','
            state'],\
196                 axis = 1, inplace = True)
197     mal_vis_rank['tol_visitor'] = mal_vis_rank['tol_visitor'].astype(str)
198     mal_vis_rank_json = mal_vis_rank.to_json(orient='records')
199     mal_vis_rank_json = json.loads(mal_vis_rank_json)
200     mal_rank['mal_vis_rank'] = mal_vis_rank_json
201     mal_rank['date'] = date
202 
203     return mal_rank
204 def monthlyqmsops(companyname,month,url,port,TT):
205     if TT == True:
206         mal_data,sta_data,tmp_data = getqmsttmonthlydata(companyname,month
                ,url,port)
207     else:
208         mal_data,sta_data,tmp_data = getqmsmonthlydata(companyname,month,
                url,port)
209 
210     if len(mal_data) > 0 and len(sta_data) > 0 and len(tmp_data) > 0:
211 
212         # General Processing
213         # Malaysia Operations
214 
215         # print(tmp_data)
216         weekno = ace2.datetime.today().isocalendar()[1]
217         year = ace2.datetime.today().strftime("%Y")
218         unique_key = year+month
219 
220         print(unique_key)
221 
222 
223         mal_data['mal_act_counter'] = mal_data['mal_act_counter'].astype(
                int)
224         mal_data['mal_tol_counter'] = mal_data['mal_tol_counter'].astype(
                int)
225 
226         mal_data['mal_avg_ast'] = mal_data['mal_avg_ast'].apply(lambda x:
                ace2.Mintoseconds(x))
227         mal_data['mal_avg_awt'] = mal_data['mal_avg_awt'].apply(lambda x:
                ace2.Mintoseconds(x))
```

```
228         mal_data['mal_tol_ser'] = mal_data['mal_tol_ser'].astype(int)
229         mal_data['mal_tol_visitor'] = mal_data['mal_tol_visitor'].astype(
                int)
230
231
232         mal_data_cal = dict()
233         mal_data_cal['date'] = unique_key
234         mal_data_cal['mal_act_counter'] = round(mal_data['mal_act_counter'
                ].mean())
235         mal_data_cal['mal_tol_counter'] = round(mal_data['mal_tol_counter'
                ].mean())
236         mal_data_cal['mal_avg_ast'] = ace2.secondstoMin(sum(mal_data['
                mal_avg_ast'])/len(mal_data['mal_avg_ast']))
237         mal_data_cal['mal_avg_awt'] = ace2.secondstoMin(sum(mal_data['
                mal_avg_awt'])/len(mal_data['mal_avg_awt']))
238         mal_data_cal['mal_tol_ser'] = sum(mal_data['mal_tol_ser'])
239         mal_data_cal['mal_tol_visitor'] = sum(mal_data['mal_tol_visitor'])
240         mal_data_cal['mal_buttons'] = buttonsOps(mal_data['mal_buttons'])
241
242         datelist = list()
243         mal_avg_detail = list()
244         for index,each in mal_data.iterrows():
245             data = dict()
246             data['mal_avg_ast'] = ace2.secondstoMin(each['mal_avg_ast'])
247             data['mal_avg_awt'] = ace2.secondstoMin(each['mal_avg_awt'])
248             data['date'] = each['date']
249             datelist.append(each['date'])
250             mal_avg_detail.append(data)
251
252         mal_data_cal['mal_avg_detail'] = mal_avg_detail
253
254         # States Operations
255         sta_data_group = sta_data.groupby(by=sta_data.sta_name,as_index=
                False)
256         sta_data_detail = list()
257
258         for key,value in sta_data_group:
259             group = pd.DataFrame(value)
260
261             sta_data_cal = dict()
262             sta_data_cal['sta_name'] = group.iloc[0]['sta_name']
263             group['sta_avg_ast'] = group['sta_avg_ast'].apply(lambda x:
                    ace2.Mintoseconds(x))
264             group['sta_avg_awt'] = group['sta_avg_awt'].apply(lambda x:
                    ace2.Mintoseconds(x))
265             group['sta_act_counter'] = group['sta_act_counter'].astype(int
                    )
266             group['sta_tol_counter'] = group['sta_tol_counter'].astype(int
                    )
267             group['sta_tol_ser'] = group['sta_tol_ser'].astype(int)
268             group['sta_tol_visitor'] = group['sta_tol_visitor'].astype(int
                    )
269             sta_data_cal['sta_avg_ast'] = ace2.secondstoMin(sum(group['
                    sta_avg_ast'])/len(group['sta_avg_ast']))
```

```
270             sta_data_cal['sta_avg_awt'] = ace2.secondstoMin(sum(group['
                    sta_avg_awt'])/len(group['sta_avg_awt']))
271             sta_data_cal['sta_tol_visitor'] = sum(group['sta_tol_visitor'
                    ])
272             sta_data_cal['sta_tol_ser'] = sum(group['sta_tol_ser'])
273             sta_data_cal['sta_buttons'] = buttonsOps(group['sta_buttons'])
274             sta_data_cal['sta_act_counter'] = round(group['sta_act_counter
                    '].mean())
275             sta_data_cal['sta_tol_counter'] = round(group['sta_tol_counter
                    '].mean())
276             if 'sta_counter_details' in group.columns:
277                 sta_counter_details_pd = pd.DataFrame()
278                 sta_counter_details = pd.DataFrame(group['
                        sta_counter_details'])
279                 for index,each in sta_counter_details.iterrows():
280                     if type(each['sta_counter_details']) is float:
281                         continue
282                     else:
283                         df = pd.DataFrame(each['sta_counter_details'])
284                         if len(sta_counter_details_pd) < 0 :
285                             sta_counter_details_pd = df
286                         else:
287                             sta_counter_details_pd =
                                    sta_counter_details_pd.append(df,
                                    ignore_index=True)
288                 sta_counter_details_group = sta_counter_details_pd.groupby
                        (by=sta_counter_details_pd.premisename,\
289                                                  as_index=False)
290                 sta_counter_details = list()
291                 for key,each in sta_counter_details_group:
292                     data = dict()
293                     data['premisename'] = each.iloc[0]['premisename']
294                     data['activeCounter'] = max(each['activeCounter'])
295                     data['totalCounter'] = max(each['totalCounter'])
296                     sta_counter_details.append(data)
297                 sta_data_cal['sta_counter_details'] = sta_counter_details
298
299
300             sta_avg_detail = list()
301             count = 0
302             for index,each in group.iterrows():
303
304                 data = dict()
305                 data['sta_avg_ast'] = ace2.secondstoMin(each['sta_avg_ast'
                        ])
306                 data['sta_avg_awt'] = ace2.secondstoMin(each['sta_avg_awt'
                        ])
307                 data['date'] = datelist[count]
308                 count = count + 1
309                 sta_avg_detail.append(data)
310             sta_data_cal['sta_avg_detail'] = sta_avg_detail
311             sta_data_detail.append(sta_data_cal)
312         mal_data_cal['state_detail'] = sta_data_detail
313
```

```
314
315
316         # TMpoints Operations
317         tmpoints_group = tmp_data.groupby(by=tmp_data.premisename,as_index
                =False)
318         tmp_detail = list()
319
320         for key,value in tmpoints_group:
321             group = pd.DataFrame(value)
322
323             tmpoint_cal = dict()
324             tmpoint_cal['premisename'] = group.iloc[0]['premisename']
325             tmpoint_cal['state'] = group.iloc[0]['state']
326             group['serveAvg'] = group['serveAvg'].apply(lambda x: ace2.
                    Mintoseconds(x))
327             group['waitingAvg'] = group['waitingAvg'].apply(lambda x: ace2
                    .Mintoseconds(x))
328             group['tol_serve'] = group['tol_serve'].astype(int)
329             group['tol_visitor'] = group['tol_visitor'].astype(int)
330             group['activeCounter'] = group['activeCounter'].astype(int)
331             group['totalCounter'] = group['totalCounter'].astype(int)
332
333             tmpoint_cal['serveAvg'] = ace2.secondstoMin(sum(group['
                    serveAvg'])/len(group['serveAvg']))
334             tmpoint_cal['waitingAvg'] = ace2.secondstoMin(sum(group['
                    waitingAvg'])/len(group['waitingAvg']))
335             tmpoint_cal['tol_serve'] = sum(group['tol_serve'])
336             tmpoint_cal['tol_visitor'] = sum(group['tol_visitor'])
337             tmpoint_cal['buttons'] = buttonsOps(group['buttons'])
338
339             tmpoint_cal['activeCounter'] = round(group['activeCounter'].
                    mean())
340             tmpoint_cal['totalCounter'] = round(group['totalCounter'].mean
                    ())
341
342
343
344
345             counterServeAvg_detail = list()
346             for index,each in group.iterrows():
347                 data = dict()
348                 data['counterServeAvg'] = each['counterServeAvg']
349                 data['date'] = each['date']
350                 counterServeAvg_detail.append(data)
351             tmpoint_cal['counterServeAvg_detail'] = counterServeAvg_detail
352
353             avg_detail = list()
354             for index,each in group.iterrows():
355                 data = dict()
356                 data['serveAvg'] = ace2.secondstoMin(each['serveAvg'])
357                 data['waitingAvg'] = ace2.secondstoMin(each['waitingAvg'])
358                 data['date'] = each['date']
359                 avg_detail.append(data)
360             tmpoint_cal['avg_detail'] = avg_detail
```

```
361
362             tmp_detail.append(tmpoint_cal)
363
364         mal_data_cal['tmpoint_detail'] = tmp_detail
365
366         #  Ranking Processing
367         tmp_detail_pd = pd.DataFrame.from_dict(tmp_detail)
368         mal_rank = calculaterank(tmp_detail_pd,unique_key)
369
370         if TT == True:
371             postqmsttmonthly(mal_data_cal,companyname,url,port)
372         else:
373
374             postqmsmonthly(mal_data_cal,companyname,url,port)
375             postqmsmonthlyrank(mal_rank,companyname,url,port)
376
377
378 def main(month, TT = False):
379     source = ace2.read_json('../input.json')
380     url = source['url']
381     port = source['port']
382     companydetails = ace2.pickpremisebycom(url,port)
383     for each in companydetails:
384         monthlyqmsops(each['name'],month,url,port,TT)
```

ace2/qms/weekly.py

```
1  import ace2
2  import os.path
3  import pandas as pd
4  import json
5  import requests as rq
6
7  # QMS
8  def getqmsweeklydata(companyname,url,port,weekno):
9      request = rq.get ('http://'+url+':'+port+'/ace/api/v1/qms/weeklydata?
           company='+companyname+'&week='+weekno, \
10             headers= {'Content-type': 'application/json'})
11     value = request.text
12     data = json.loads(value)
13     if data['error'] == False:
14         data = data['QMSWeeklyData']
15         if len(data) == 0:
16
17             tmp_data = pd.DataFrame()
18             sta_data = pd.DataFrame()
19             mal_data = pd.DataFrame()
20             return (mal_data,sta_data,tmp_data)
21         else:
22             columns_flag = False
23             for each in data:
24                 dailydata = each['data']
```

```
                dailytmpoints = dailydata['tmpoint_detail']
                dailystates = dailydata['state_detail']
                dailystates = pd.io.json.json_normalize(dailystates)
                dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
                dailymal = pd.io.json.json_normalize(dailydata)
                dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                    inplace = True)
                if columns_flag == False:
                    tmp_data = dailytmpoints
                    sta_data = dailystates
                    mal_data = dailymal
                    columns_flag= True
                else:

                    tmp_data = tmp_data.append(dailytmpoints, ignore_index
                        =True)
                    sta_data = sta_data.append(dailystates, ignore_index=
                        True)
                    mal_data = mal_data.append(dailymal, ignore_index=True
                        )
            return (mal_data,sta_data,tmp_data)
def getqmspreweeklydata(companyname,url,port,weekno):
    request = rq.get ('http://'+url+':'+port+'/ace/api/v1/qms/
        previousweekdata?company='+companyname+'&week='+weekno, \
            headers= {'Content-type': 'application/json'})

    value = request.text
    data = json.loads(value)
    if data['error'] == False:
        data = data['QMSPreviousWeekData']
        if len(data) == 0:

            tmp_data = pd.DataFrame()
            sta_data = pd.DataFrame()
            mal_data = pd.DataFrame()
            return (mal_data,sta_data,tmp_data)
        else:
            columns_flag = False
            for each in data:
                dailydata = each['data']
                dailytmpoints = dailydata['tmpoint_detail']
                dailystates = dailydata['state_detail']
                date = dailydata['date']
                dailystates = pd.io.json.json_normalize(dailystates)
                dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
                dailytmpoints['date'] = date
                dailymal = pd.io.json.json_normalize(dailydata)
                dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                    inplace = True)
                if columns_flag == False:
                    tmp_data = dailytmpoints
                    sta_data = dailystates
                    mal_data = dailymal
                    columns_flag= True
```

```
73                  else:
74
75                      tmp_data = tmp_data.append(dailytmpoints, ignore_index
                            =True)
76                      sta_data = sta_data.append(dailystates, ignore_index=
                            True)
77                      mal_data = mal_data.append(dailymal, ignore_index=True
                            )
78              return (mal_data,sta_data,tmp_data)
79  def postqmsweekly(data,companyname,url,port):
80      url = 'http://'+url+':'+port+'/ace/api/v1/qms/weekly?company='+
            companyname
81      headers = {'Content-type': 'application/json'}
82      r = rq.post(url, data=json.dumps(data), headers=headers)
83      print(r.text)
84  def postqmsweeklyrank(data,companyname,url,port):
85      url = 'http://'+url+':'+port+'/ace/api/v1/qms/weeklyrank?company='+
            companyname
86      headers = {'Content-type': 'application/json'}
87      r = rq.post(url, data=json.dumps(data), headers=headers)
88      print(r.text)
89
90  # QMS TT
91  def getqmsttweeklydata(companyname,url,port,weekno):
92      request = rq.get ('http://'+url+':'+port+'/ace/api/v1/qmstt/weeklydata
            ?company='+companyname+'&week='+weekno, \
93              headers= {'Content-type': 'application/json'})
94      value = request.text
95      data = json.loads(value)
96      if data['error'] == False:
97          data = data['QMSttWeeklyData']
98          if len(data) == 0:
99
100             tmp_data = pd.DataFrame()
101             sta_data = pd.DataFrame()
102             mal_data = pd.DataFrame()
103             return (mal_data,sta_data,tmp_data)
104         else:
105             columns_flag = False
106             for each in data:
107                 dailydata = each['data']
108                 dailytmpoints = dailydata['tmpoint_detail']
109                 dailystates = dailydata['state_detail']
110                 dailystates = pd.io.json.json_normalize(dailystates)
111                 dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
112                 dailymal = pd.io.json.json_normalize(dailydata)
113                 dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                        inplace = True)
114                 if columns_flag == False:
115                     tmp_data = dailytmpoints
116                     sta_data = dailystates
117                     mal_data = dailymal
118                     columns_flag= True
119                 else:
```

```
120
121                     tmp_data = tmp_data.append(dailytmpoints, ignore_index
                            =True)
122                     sta_data = sta_data.append(dailystates, ignore_index=
                            True)
123                     mal_data = mal_data.append(dailymal, ignore_index=True
                            )
124             return (mal_data,sta_data,tmp_data)
125 def getqmsttpreweeklydata(companyname,url,port,weekno):
126     request = rq.get ('http://'+url+':'+port+'/ace/api/v1/qmstt/
            previousweekdata?company='+companyname+'&week='+weekno, \
127             headers= {'Content-type': 'application/json'})
128
129     value = request.text
130     data = json.loads(value)
131     if data['error'] == False:
132         data = data['QMSttPreviousWeekData']
133         if len(data) == 0:
134
135             tmp_data = pd.DataFrame()
136             sta_data = pd.DataFrame()
137             mal_data = pd.DataFrame()
138             return (mal_data,sta_data,tmp_data)
139         else:
140             columns_flag = False
141             for each in data:
142                 dailydata = each['data']
143                 dailytmpoints = dailydata['tmpoint_detail']
144                 dailystates = dailydata['state_detail']
145                 date = dailydata['date']
146                 dailystates = pd.io.json.json_normalize(dailystates)
147                 dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
148                 dailytmpoints['date'] = date
149                 dailymal = pd.io.json.json_normalize(dailydata)
150                 dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                        inplace = True)
151                 if columns_flag == False:
152                     tmp_data = dailytmpoints
153                     sta_data = dailystates
154                     mal_data = dailymal
155                     columns_flag= True
156                 else:
157
158                     tmp_data = tmp_data.append(dailytmpoints, ignore_index
                            =True)
159                     sta_data = sta_data.append(dailystates, ignore_index=
                            True)
160                     mal_data = mal_data.append(dailymal, ignore_index=True
                            )
161             return (mal_data,sta_data,tmp_data)
162 def postqmsttweekly(data,companyname,url,port):
163     url = 'http://'+url+':'+port+'/ace/api/v1/qmstt/weekly?company='+
            companyname
164     headers = {'Content-type': 'application/json'}
```

```
165     r = rq.post(url, data=json.dumps(data), headers=headers)
166     print(r.text)
167 
168 # Functional Operations
169 def buttonsOps(buttons) :
170     # print(buttons)
171     a_list = list()
172     for i in range(len(buttons)):
173         a = list(buttons.iloc[i])
174         a_list.append(a)
175 
176     a_df = pd.DataFrame(a_list)
177 
178     final_sum = list()
179     for i in range(len(a_df.columns)):
180         a_col_list = list(a_df.ix[:,i])
181         new_dictionary = {}
182         for dictionary in a_col_list:
183             if dictionary is not None:
184                 for key, value in dictionary.items():
185                     if key in new_dictionary.keys():
186                         if key == 'value':
187                             new_dictionary[key] = int(value) + int(
                                    new_dictionary[key])
188                         else:
189                             new_dictionary[key] = new_dictionary[key]
190                     else:
191                         new_dictionary[key] = value
192 
193 
194 
195         final_sum.append(new_dictionary)
196 
197     return final_sum
198 def counterSeverAvgOps(counterServeAvg) :
199     a_list = list()
200     for i in range(len(counterServeAvg)):
201         a = list(counterServeAvg.iloc[i])
202         a_list.append(a)
203 
204     a_df = ace2.pd.DataFrame(a_list)
205 
206     final_sum = list()
207     for i in range(len(a_df.columns)):
208         a_col_list = list(a_df.ix[:,i])
209         new_dictionary = {}
210         for dictionary in a_col_list:
211             for key, value in dictionary.items():
212                 if key in new_dictionary.keys():
213                     if key == 'value':
214                         new_dictionary[key] =   ace2.Mintoseconds(value) +
                                new_dictionary[key]
215                     if key == 'name':
216                         new_dictionary[key] = value
```

```
217                 else:
218                     if key == 'value':
219                         new_dictionary[key] = ace2.Mintoseconds(value)
220                     if key == 'name':
221                         new_dictionary[key] = value
222         for key, value in new_dictionary.items():
223             if key == 'value':
224                 new_dictionary[key] = ace2.secondstoMin(value/len(a_df))
225 
226 
227         final_sum.append(new_dictionary)
228     return final_sum
229 def calculaterank(tmpoints,date):
230     # Daily Rank Data
231     # State Operations
232     sta_rank_group = tmpoints.groupby(by=tmpoints.state,as_index=False)
233     sta_rank_detail = list()
234     for key,value in sta_rank_group:
235         group = pd.DataFrame(value)
236         state = dict()
237         state['sta_name'] = group.iloc[0]['state']
238         ser_rank = group.sort_values(by=['tol_serve'], ascending=False)
239         ser_rank.drop(['tol_visitor','buttons','serveAvg','waitingAvg','
               state'],\
240                     axis = 1, inplace = True)
241         ser_rank['tol_serve'] = ser_rank['tol_serve'].astype(str)
242         ser_rank_json = ser_rank.to_json(orient='records')
243         ser_rank_json = json.loads(ser_rank_json)
244         state['sta_ser_rank'] = ser_rank_json
245 
246 
247         cur_rank = group.sort_values(by=['tol_visitor'], ascending=False)
248         cur_rank.drop(['tol_serve','buttons','serveAvg','waitingAvg','
               state'],\
249                     axis = 1, inplace = True)
250         cur_rank['tol_visitor'] = cur_rank['tol_visitor'].astype(str)
251         cur_rank_json = cur_rank.to_json(orient='records')
252         cur_rank_json = json.loads(cur_rank_json)
253         state['sta_vis_rank'] = cur_rank_json
254         sta_rank_detail.append(state)
255 
256     mal_ser_rank = tmpoints.sort_values(by=['tol_serve'], ascending=False)
257     mal_ser_rank.drop(['tol_visitor','buttons','serveAvg','waitingAvg','
            state'],\
258                 axis = 1, inplace = True)
259 
260     # Rank data Processing
261     mal_rank = dict()
262     mal_rank['state_rank'] =  sta_rank_detail
263     mal_ser_rank['tol_serve'] = mal_ser_rank['tol_serve'].astype(str)
264     mal_ser_rank_json = mal_ser_rank.to_json(orient='records')
265     mal_ser_rank_json = json.loads(mal_ser_rank_json)
266     mal_rank['mal_ser_rank'] = mal_ser_rank_json
267 
```

```
268     mal_vis_rank = tmpoints.sort_values(by=['tol_visitor'], ascending=
            False)
269     mal_vis_rank.drop(['tol_serve','buttons','serveAvg','waitingAvg','
            state'],\
270                       axis = 1, inplace = True)
271     mal_vis_rank['tol_visitor'] = mal_vis_rank['tol_visitor'].astype(str)
272     mal_vis_rank_json = mal_vis_rank.to_json(orient='records')
273     mal_vis_rank_json = json.loads(mal_vis_rank_json)
274     mal_rank['mal_vis_rank'] = mal_vis_rank_json
275     mal_rank['date'] = date
276 
277     return mal_rank
278 def weeklyqmsops(companyname,url,port,weekno,pre,TT):
279     if pre == True:
280         if TT == True:
281             mal_data,sta_data,tmp_data = getqmsttpreweeklydata(companyname
                    ,url,port,weekno)
282         else:
283             mal_data,sta_data,tmp_data = getqmspreweeklydata(companyname,
                    url,port,weekno)
284     else:
285         if TT == True:
286             mal_data,sta_data,tmp_data = getqmsttweeklydata(companyname,
                    url,port,weekno)
287         else:
288             mal_data,sta_data,tmp_data = getqmsweeklydata(companyname,url,
                    port,weekno)
289 
290     if len(mal_data) > 0 and len(sta_data) > 0 and len(tmp_data) > 0:
291 
292         # General Processing
293         # Malaysia Operations
294 
295         # print(tmp_data)
296         year = ace2.datetime.today().strftime("%Y")
297         unique_key = year+'-'+str(weekno)
298         print(unique_key)
299 
300         mal_data['mal_act_counter'] = mal_data['mal_act_counter'].astype(
                int)
301         mal_data['mal_tol_counter'] = mal_data['mal_tol_counter'].astype(
                int)
302 
303         mal_data['mal_avg_ast'] = mal_data['mal_avg_ast'].apply(lambda x:
                ace2.Mintoseconds(x))
304         mal_data['mal_avg_awt'] = mal_data['mal_avg_awt'].apply(lambda x:
                ace2.Mintoseconds(x))
305         mal_data['mal_tol_ser'] = mal_data['mal_tol_ser'].astype(int)
306         mal_data['mal_tol_visitor'] = mal_data['mal_tol_visitor'].astype(
                int)
307         count = 0
308         for each in  list(mal_data['mal_avg_ast']):
309             if each > 0:
310                 count += 1
```

```
311         if count == 0:
312             count = 1
313         mal_data_cal = dict()
314         mal_data_cal['date'] = unique_key
315         mal_data_cal['mal_act_counter'] = round(mal_data['mal_act_counter'
                ].mean())
316         mal_data_cal['mal_tol_counter'] = round(mal_data['mal_tol_counter'
                ].mean())
317         mal_data_cal['mal_avg_ast'] = ace2.secondstoMin(round(sum(mal_data
                ['mal_avg_ast'])/count))
318         mal_data_cal['mal_avg_awt'] = ace2.secondstoMin(round(sum(mal_data
                ['mal_avg_awt'])/count))
319         mal_data_cal['mal_tol_ser'] = sum(mal_data['mal_tol_ser'])
320         mal_data_cal['mal_tol_visitor'] = sum(mal_data['mal_tol_visitor'])
321         mal_data_cal['mal_buttons'] = buttonsOps(mal_data['mal_buttons'])
322 
323         datelist = list()
324         mal_avg_detail = list()
325         for index,each in mal_data.iterrows():
326             data = dict()
327             data['mal_avg_ast'] = ace2.secondstoMin(each['mal_avg_ast'])
328             data['mal_avg_awt'] = ace2.secondstoMin(each['mal_avg_awt'])
329             data['date'] = each['date']
330             datelist.append(each['date'])
331             mal_avg_detail.append(data)
332 
333         mal_data_cal['mal_avg_detail'] = mal_avg_detail
334 
335 
336 
337 
338         # States Operations
339         sta_data_group = sta_data.groupby(by=sta_data.sta_name,as_index=
                False)
340         sta_data_detail = list()
341 
342         for key,value in sta_data_group:
343             group = pd.DataFrame(value)
344 
345             sta_data_cal = dict()
346             sta_data_cal['sta_name'] = group.iloc[0]['sta_name']
347             group['sta_avg_ast'] = group['sta_avg_ast'].apply(lambda x:
                    ace2.Mintoseconds(x))
348             group['sta_avg_awt'] = group['sta_avg_awt'].apply(lambda x:
                    ace2.Mintoseconds(x))
349             group['sta_act_counter'] = group['sta_act_counter'].astype(int
                    )
350             group['sta_tol_counter'] = group['sta_tol_counter'].astype(int
                    )
351             group['sta_tol_ser'] = group['sta_tol_ser'].astype(int)
352             group['sta_tol_visitor'] = group['sta_tol_visitor'].astype(int
                    )
353 
354             count = 0
```

```
355             for each in  list(group['sta_avg_ast']):
356                 if each > 0:
357                     count += 1
358             if count == 0:
359                 count = 1
360 
361             sta_data_cal['sta_avg_ast'] = ace2.secondstoMin(round(sum(
                    group['sta_avg_ast'])/count))
362             sta_data_cal['sta_avg_awt'] = ace2.secondstoMin(round(sum(
                    group['sta_avg_awt'])/count))
363             sta_data_cal['sta_tol_visitor'] = sum(group['sta_tol_visitor'
                    ])
364             sta_data_cal['sta_tol_ser'] = sum(group['sta_tol_ser'])
365             sta_data_cal['sta_buttons'] = buttonsOps(group['sta_buttons'])
366             sta_data_cal['sta_act_counter'] = round(group['sta_act_counter
                    '].mean())
367             sta_data_cal['sta_tol_counter'] = round(group['sta_tol_counter
                    '].mean())
368 
369 
370             if 'sta_counter_details' in group.columns:
371                 sta_counter_details_pd = pd.DataFrame()
372                 sta_counter_details = pd.DataFrame(group['
                        sta_counter_details'])
373                 for index,each in sta_counter_details.iterrows():
374                     if type(each['sta_counter_details']) is float:
375                         continue
376                     else:
377                         df = pd.DataFrame(each['sta_counter_details'])
378                         if len(sta_counter_details_pd) < 0 :
379                             sta_counter_details_pd = df
380                         else:
381                             sta_counter_details_pd =
                                    sta_counter_details_pd.append(df,
                                    ignore_index=True)
382 
383                 sta_counter_details_group = sta_counter_details_pd.groupby
                        (by=sta_counter_details_pd.premisename,\
384                                                     as_index=False)
385                 sta_counter_details = list()
386                 for key,each in sta_counter_details_group:
387                     data = dict()
388                     data['premisename'] = each.iloc[0]['premisename']
389                     data['activeCounter'] = max(each['activeCounter'])
390                     data['totalCounter'] = max(each['totalCounter'])
391                     sta_counter_details.append(data)
392                 sta_data_cal['sta_counter_details'] = sta_counter_details
393 
394 
395             sta_avg_detail = list()
396             count = 0
397             for index,each in group.iterrows():
398                 data = dict()
```

```
                data['sta_avg_ast'] = ace2.secondstoMin(each['sta_avg_ast'])
                data['sta_avg_awt'] = ace2.secondstoMin(each['sta_avg_awt'])
                data['date'] = datelist[count]
                count = count + 1
                sta_avg_detail.append(data)
            sta_data_cal['sta_avg_detail'] = sta_avg_detail
            sta_data_detail.append(sta_data_cal)
        mal_data_cal['state_detail'] = sta_data_detail



        # TMpoints Operations
        tmpoints_group = tmp_data.groupby(by=tmp_data.premisename,as_index=False)
        tmp_detail = list()

        for key,value in tmpoints_group:
            group = pd.DataFrame(value)

            tmpoint_cal = dict()
            tmpoint_cal['premisename'] = group.iloc[0]['premisename']


            tmpoint_cal['state'] = group.iloc[0]['state']
            group['serveAvg'] = group['serveAvg'].apply(lambda x: ace2.Mintoseconds(x))
            group['waitingAvg'] = group['waitingAvg'].apply(lambda x: ace2.Mintoseconds(x))
            group['tol_serve'] = group['tol_serve'].astype(int)
            group['tol_visitor'] = group['tol_visitor'].astype(int)
            group['activeCounter'] = group['activeCounter'].astype(int)
            group['totalCounter'] = group['totalCounter'].astype(int)
            count = 0
            for each in  list(group['waitingAvg']):
                if each > 0:
                    count += 1
            if count == 0:
                count = 1
            tmpoint_cal['serveAvg'] = ace2.secondstoMin(round(sum(group['serveAvg'])/count))
            tmpoint_cal['waitingAvg'] = ace2.secondstoMin(round(sum(group['waitingAvg'])/count))
            tmpoint_cal['tol_serve'] = sum(group['tol_serve'])
            tmpoint_cal['tol_visitor'] = sum(group['tol_visitor'])
            tmpoint_cal['buttons'] = buttonsOps(group['buttons'])
            tmpoint_cal['activeCounter'] = round(group['activeCounter'].mean())
            tmpoint_cal['totalCounter'] = round(group['totalCounter'].mean())


            counterServeAvg_detail = list()
```

```
444                 for index,each in group.iterrows():
445                     data = dict()
446                     data['counterServeAvg'] = each['counterServeAvg']
447                     data['date'] = each['date']
448                     counterServeAvg_detail.append(data)
449                 tmpoint_cal['counterServeAvg_detail'] = counterServeAvg_detail
450 
451                 avg_detail = list()
452                 for index,each in group.iterrows():
453                     data = dict()
454                     data['serveAvg'] = ace2.secondstoMin(each['serveAvg'])
455                     data['waitingAvg'] = ace2.secondstoMin(each['waitingAvg'])
456                     data['date'] = each['date']
457                     avg_detail.append(data)
458                 tmpoint_cal['avg_detail'] = avg_detail
459 
460                 tmp_detail.append(tmpoint_cal)
461 
462             mal_data_cal['tmpoint_detail'] = tmp_detail
463 
464             #  Ranking Processing
465             tmp_detail_pd = pd.DataFrame.from_dict(tmp_detail)
466             mal_rank = calculaterank(tmp_detail_pd,unique_key)
467 
468 
469             if TT == True:
470                 postqmsttweekly(mal_data_cal,companyname,url,port)
471             else:
472                 postqmsweekly(mal_data_cal,companyname,url,port)
473                 postqmsweeklyrank(mal_rank,companyname,url,port)
474         else:
475             print('NO Week No.',weekno,'Data')
476 
477 
478 def main(weekno,pre=False,TT=False):
479     source = ace2.read_json('../input.json')
480     url = source['url']
481     port = source['port']
482     companydetails = ace2.pickpremisebycom(url,port)
483     for each in companydetails:
484         weeklyqmsops(each['name'],url,port,weekno,pre,TT)
```

ace2/target/___init___.py

```
1 __all__ = ['process', 'target']
2 
3 import ace2.target.process
4 import ace2.target.target
```

ace2/target/process.py

```
import ace2
import os.path
import pandas as pd
import json
import functools as ft
import requests as rq


# Checking Ops
def checkage(key):

    if key == '>21':
        key = 'age21'
    elif key == '18-21':
        key = 'age1821'
    elif key == '<18':
        key = 'age18'
    else:
        key = ''
    return key


def checkgender(key):
    if key == 'f':
        key = 'female'
    elif key == 'm':
        key = 'male'
    else:
        key = ''
    return key


def checkProd(x):
    unifilite = ['UniFiLite10MbpsSUBBBundle', 'UniFiLite10MbpsHSBB']
    unifibiz = ['UniFiBizLite10MbpsSUBBBundle', 'UniFiBizLite10MbpsHSBB',
                'UniFiBizPro100Mbps', 'UniFiBizAdvance30Mbps']
    unifiadvance = ['UniFiAdvance30Mbps', 'UniFiAdvancePlus50Mbps', '
       UniFiPro100Mbps']
    streamyx_10 = ['StreamyxBlockbusterCombo10MBundle']
    streamyx_20 = ['StreamyxBlockbusterCombo20MBundle', '
       StreamyxSOHOOIAB20MmeBundle']
    streamyx_40 = ['StreamyxSOHOOIAB40MmeBundle', '
       StreamyxBlockbusterCombo40MBundle']
    streamyx_80 = ['StreamyxBlockbusterCombo80MHyppTVBundle']
    streamyx_soho = ['StreamyxSOHOOIAB80MmeBundle', '
       StreamyxSOHOOIAB10MmeBundle']
    if x in unifilite:
        return 'unifilite'
    elif x in unifibiz:
        return 'unifibiz'
    elif x in unifiadvance:
        return 'unifiadvance'
    elif x in streamyx_10:
```

```
        return 'streamyx_1'
    elif x in streamyx_20:
        return 'streamyx_2'
    elif x in streamyx_40:
        return 'streamyx_4'
    elif x in streamyx_80:
        return 'streamyx_8'
    elif x in streamyx_soho:
        return 'streamyx_soho'
    else:
        return 'unifilite'


def checkGenderFromID(x):
    if x is not None:
        if len(x) == 14:
            lastdigit = x[13]
            try:
                if int(lastdigit) % 2 == 0:
                    return('female')
                else:
                    return('male')
            except Exception as e:
                return('')

        else:
            return('')
    else:
        return('')


def checkageFromDOB(x):
    if x is not None:
        if len(x) > 4:
            year = x[:4]
            thisyear = ace2.datetime.now().strftime("%Y")
            age = int(thisyear)-int(year)
            if age > 21:
                return ('age21')
            elif age < 18:
                return('age18')
            elif age < 21 and age > 18:
                return('age1821')
            else:
                return ('')
        else:
            return ('')
    else:
        return ('')


def checkstate(x):
    if x == 0:
        return('')
```

```
104     else:
105         try:
106             x = x.lower()
107             return x
108         except Exception as e:
109             return('')
110 
111 
112 def gettargetalldata_captive(url, port, company, weekno, weekflag):
113     '''get target Data from captive portal'''
114     if weekflag == False:
115         url = 'http://'+url+':'+port+'/ace/api/target/all?company='+
                company
116         name = 'TargetedMarketingAll'
117     else:
118         url = 'http://'+url+':'+port+'/ace/api/target/weekly?company='+
                company +\
119             '&week='+weekno
120         name = 'TargetedMarketingWeekly'
121     request = ace2.rq.get(url, headers={'Content-type': 'application/json'
            })
122     value = request.text
123 
124     targetdata = ace2.json.loads(value)
125     error_code = True
126     if targetdata['error'] == False:
127         targetdata = targetdata[name]
128         if len(targetdata) == 0:
129             df = pd.DataFrame()
130             return (df, error_code)
131         else:
132             final_data = pd.DataFrame(targetdata)
133             final_data['gender'] = final_data['gender'].apply(lambda x:
                    checkgender(x))
134             final_data['age_range'] = final_data['age_range'].apply(lambda
                     x: checkage(x))
135             error_code = False
136             return (final_data, error_code)
137     else:
138         df = pd.DataFrame()
139         return (df, error_code)
140 
141 
142 def gettargetalldata_dice(url, port, company, date):
143     ''' get target Data from captive portal  '''
144 
145     url = 'http://'+url+':'+port+'/ace/api/dice/order?date='+date+'&
            company='+company
146     request = ace2.rq.get(url, headers={'Content-type': 'application/json'
            })
147     value = request.text
148 
149     targetdata = ace2.json.loads(value)
150     error_code = True
```

```
151     if targetdata['error'] == False:
152         targetdata = targetdata['OrderData']
153         if len(targetdata) == 0:
154             df = pd.DataFrame()
155             return (df, error_code)
156         else:
157             targetdata = targetdata[0]
158             data = targetdata['data']
159             final_data = pd.DataFrame(data)
160             error_code = False
161             return (final_data, error_code)
162     else:
163         df = pd.DataFrame()
164         return (df, error_code)
165 
166 
167 def postemail(data, url, port, company):
168     url = 'http://'+url+':'+port+'/ace/api/target/user?company='+company
169     headers = {'Content-type': 'application/json'}
170     r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
171     print('Result: ', r.text)
172 
173 
174 def main(date, data_flag, weekno='00', weekflag=False):
175     source = ace2.read_json('../input.json')
176     url = source['url']
177     port = source['port']
178     companydetails = ace2.pickpremisebycom(url, port)
179     for each in companydetails:
180         if data_flag == 'captive':
181             data, error_code = gettargetalldata_captive(url, port, each['
                name'],
182                                                         weekno, weekflag)
183             if error_code == False:
184                 data = data[['premisename', 'age_range', 'email', 'gender'
                    ]]
185                 premisedata = ace2.getpremises(url, port)
186                 data = pd.merge(data, premisedata, on='premisename', how='
                    left')
187                 data = data.fillna(0)
188                 data['state'] = data['state'].apply(lambda x: checkstate(x
                    ))
189                 data = data.drop(['premisename'], axis=1)
190                 data['package'] = data['state'].apply(lambda x: '')
191                 data = data.rename(index=str, columns={'state': 'location'
                    ,
192                                                        'age_range': 'age'
                                                           })
193                 data_json = json.loads(data.to_json(orient='records'))
194                 for item in data_json:
195                     postemail(item, url, port, each['name'])
196 
197             else:
```

```
198                 print('Problem encounter in Get Target Data or No Data
                        belong to ', each['name'])
199         elif data_flag == 'dice':
200             data, error_code = gettargetalldata_dice(url, port, each['name
                    '], date)
201             if error_code == False and len(data) > 0:
202                 print('Running for ', each['name'])
203 
204                 # Product
205                 data['product'] = data['product'].apply(
206                     lambda x: ''.join(e for e in x if e.isalnum()))
207                 state_uniq_list = list(set(list(data['state'])))
208                 data['product'] = data['product'].apply(lambda x:
                        checkProd(x))
209                 data['dob'] = data['dob'].apply(lambda x: checkageFromDOB(
                        x))
210 
211                 data['idvalue'] = data['idvalue'].apply(lambda x:
                        checkGenderFromID(x))
212                 data['state'] = data['state'].apply(lambda x: checkstate(x
                        ))
213                 data = data[['email', 'state', 'product', 'dob', 'idvalue'
                        ]]
214 
215                 data = data.rename(index=str, columns={'state': 'location'
                        ,
216                                                        'product': 'package
                                                            ',
217                                                        'dob': 'age',
218                                                        'idvalue': 'gender'
                                                            })
219                 # print(data)
220                 data_json = json.loads(data.to_json(orient='records'))
221 
222                 for item in data_json:
223                     postemail(item, url, port, each['name'])
224 
225             else:
226                 print('No Data for company', each['name'])
```

ace2/target/target.py

```
1  import ace2
2  import os.path
3  import pandas as pd
4  import json
5  import requests as rq
6  from datetime import datetime
7  from collections import Counter
8 
9 
10 def gettargetallda(url, port, company):
```

```
    '''get target Data from target'''

    url = 'http://'+url+':'+port+'/ace/api/target/data?company='+company
    request = ace2.rq.get(url, headers={'Content-type': 'application/json'
        })
    value = request.text

    targetdata = ace2.json.loads(value)
    error_code = True
    if targetdata['error'] == False:
        targetdata = targetdata['data']
        try:
            promo = pd.DataFrame(targetdata['promo'])
            hitcount = pd.DataFrame(targetdata['hitcount'])
            if promo.empty and hitcount.empty:
                return (promo, hitcount, error_code)
            else:
                error_code = False
                return (promo, hitcount, error_code)
        except Exception as e:
            promo = pd.DataFrame()
            hitcount = pd.DataFrame()
            return(promo, hitcount, error_code)
    else:
        promo = pd.DataFrame()
        hitcount = pd.DataFrame()
        return(promo, hitcount, error_code)


def transfrom_d_m(x):
    x = x.strftime("%b")
    return x


def clickandrecipient(promo, hitcount):
    """ Genegrate the total number of click and recipient
    based on Adv title.
    Args:
        promo (dataframe): promotion info DataFrame
        hitcount (dataframe): hitcount DataFrame
    Return:
        json_click_recipent(list): json object for graph

    """
    promo = promo[['title', 'recipient_no']]
    hitcount = hitcount[['title', 'hitcount']]

    new_df = hitcount.merge(promo, left_on='title', right_on='title')
    new_df = new_df.rename(columns={'hitcount': 'click', 'recipient_no': '
        recipient'})
    json_click_recipent = new_df.to_json(orient='records')
    return (json.loads(json_click_recipent))


```

```
63 def pastadvno(promo):
64     """ Genegrate the total number of adv posted in each month.
65     Args:
66         promo (dataframe): promotion info DataFrame
67 
68     Return:
69         json_pastadvno(list): json object for graph
70 
71     """
72     promo['date'] = promo['date'].apply(lambda x: datetime.strptime(x, "%Y
       %m%d"))
73     promo = promo.sort_values(by=['date'])
74     promo = list(promo['date'])
75     promo = [transfrom_d_m(x) for x in promo]
76     promo = Counter(promo)
77     json_pastadvno = list()
78     for each in promo:
79         item = dict()
80         item['month'] = each
81         item['value'] = promo[each]
82         json_pastadvno.append(item)
83     return (json_pastadvno)
84 
85 
86 def pastrecipientno(promo):
87     """ Genegrate the total number of recipient in each month.
88     Args:
89         promo (dataframe): promotion info DataFrame
90 
91     Return:
92         json_pastrecipientno(list): json object for graph
93 
94     """
95     # promo['date'] = promo['date'].apply(lambda x: datetime.strptime(x,
       "%Y%m%d"))
96     promo = promo.sort_values(by=['date'])
97     promo['date'] = promo['date'].apply(lambda x: transfrom_d_m(x))
98     promo = promo[['date', 'recipient_no']]
99 
100    promo_group = promo.groupby(by=promo.date, sort=False)
101    json_pastrecipientno = list()
102    for key, value in promo_group:
103        item = dict()
104        group = pd.DataFrame(value)
105        item['month'] = key
106        item['value'] = int(group['recipient_no'].sum())
107        json_pastrecipientno.append(item)
108    return (json_pastrecipientno)
109 
110 
111 def mosttargetgroup(promo):
112     """ Genegrate the most sent group of recipient
113     Args:
114         promo (dataframe): promotion info DataFrame
```

```
115
116     Return:
117         json_mosttargetgroup(list): json object for graph
118
119     """
120     df = pd.DataFrame()
121     data = list(promo['recipient_group'])
122     for i in range(len(data)):
123         item = pd.DataFrame(data[i])
124         df = df.append(item, ignore_index=True)
125         df = df.astype(int)
126     df = pd.DataFrame(df.sum(axis=0), columns=['value'])
127     df = df.sort_values(by=['value'], ascending=False)
128     df['name'] = df.index
129     json_mosttargetgroup = df.to_json(orient='records')
130     return (json.loads(json_mosttargetgroup))
131
132
133 def posttarget(data, url, port, company):
134     url = 'http://'+url+':'+port+'/ace/api/target/dashboard?company='+
            company
135     headers = {'Content-type': 'application/json'}
136     r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
137     print('Result: ', r.text)
138
139
140 def main():
141     source = ace2.read_json('../input.json')
142     url = source['url']
143     port = source['port']
144     companydetails = ace2.pickpremisebycom(url, port)
145     time_ = datetime.now()
146     timestamp = time_.strftime('%s')
147
148     for each in companydetails:
149         promo, hitcount, error_code = gettargetallda(url, port, each['name
                ])
150         if not error_code:
151             target = dict()
152             target['timestamp'] = timestamp
153             json_click_recipent = clickandrecipient(promo, hitcount)
154             json_pastadvno = pastadvno(promo)
155             json_pastrecipientno = pastrecipientno(promo)
156             json_mosttargetgroup = mosttargetgroup(promo)
157
158             target['clickandrecipient'] = json_click_recipent
159             target['pastadvtotalno'] = json_pastadvno
160             target['pastrecipientno'] = json_pastrecipientno
161             target['mosttargetgroup'] = json_mosttargetgroup
162             posttarget(target, url, port, each['name'])
163         else:
164             print('Data No available for ', each['name'])
```

ace2/video/daily.py

```
import requests as rq
import json
import ace2
import pandas as pd
import random


def getvideoday(date,hour,min_,url,port):
    request = rq.get ('http://'+url+':'+port+'/ace/api/v1/count?date='+date, \
        headers= {'Content-type': 'application/json'})
    value = request.text
    videodata = json.loads(value)
    if videodata['error'] == False:
        videodata = videodata['count']
        if len(videodata) == 0:
            df = pd.DataFrame()
            return df
        else:
            timestamp = date + " " + hour + ":" + min_
            date_timestamp = ace2.datetime.strptime(timestamp,"%Y%m%d %H:%M")
            str_timestamp = date_timestamp.strftime('%s')
            video_df = pd.DataFrame()
            for each in videodata:
                if each['countdata'] is not None:
                    data_video = pd.io.json.json_normalize(each['countdata'])
                    data_video['premisename'] = each['premise']
                    data_video['time_stamp'] = data_video['time_stamp'].apply(lambda x: ace2.normalizeTimeStamp(x))
                    data_video = data_video.sort_values(by=['time_stamp'])
                    test = data_video.loc[data_video['time_stamp'] == int(str_timestamp)]
                    # print(test.index.tolist())
                    if len(test.index.tolist()) > 0:
                        index = test.index.tolist()[0]
                        video_df = video_df.append(data_video[:index], ignore_index=True)
                    # else:
                    #     video_df = video_df.append(data_video, ignore_index=True)
            return video_df
def getvideo5mindata(date,url,port):
    request = rq.get ('http://'+url+':'+port+'/ace/api/v1/count?date='+date, \
        headers= {'Content-type': 'application/json'})
    value = request.text
    videodata = json.loads(value)
    if videodata['error'] == False:
        videodata = videodata['count']
        if len(videodata) == 0:
```

```
45              df = pd.DataFrame()
46              return df
47          else:
48              video_df = pd.DataFrame()
49              for each in videodata:
50                  if each['countdata'] is not None:
51                      data_video = pd.io.json.json_normalize(each['countdata
                            '])
52                      data_video['premisename'] = each['premise']
53                      data_video['time_stamp'] = data_video['time_stamp'].
                            apply(lambda x: ace2.normalizeTimeStamp(x))
54                      data_video = data_video.sort_values(by=['time_stamp'])
55
56                      video_df = video_df.append(data_video,ignore_index=
                            True)
57              return video_df
58  def getvideolastdata(date,companyname,url,port):
59      request = rq.get ('http://'+url+':'+port+'/ace/api/v1/video/lastdata?
            date='+date+'&company='+companyname, \
60      headers= {'Content-type': 'application/json','Authorization':'1
            kqq0ltbcd5qvkd9kunjp3kfl2'})
61      value = request.text
62      value = json.loads(value)
63      if value['error'] == False:
64          predata = value['videoLastData']
65          if len(predata) == 0:
66              df_mal = pd.DataFrame()
67              df_state = pd.DataFrame()
68              df_tmpoint = pd.DataFrame()
69              return (df_mal,df_state,df_tmpoint)
70          else:
71              predata = predata[0]
72              predata = predata['data']
73              # predata = [data for data in predata if data['companyname']
                    == companyname][0]
74              # predata = predata['data']
75              mal_predata = ace2.pd.io.json.json_normalize(predata)
76              mal_predata.drop(['tmpoint_detail','state_detail'], axis = 1,
                    inplace = True)
77              state_predata = predata['state_detail']
78              state_predata = ace2.pd.io.json.json_normalize(state_predata)
79              tmpoint_predata = predata['tmpoint_detail']
80              tmpoint_predata = ace2.pd.io.json.json_normalize(
                    tmpoint_predata)
81              return (mal_predata,state_predata,tmpoint_predata)
82
83  def postvideodaily(data,company,url,port):
84      url = 'http://'+url+':'+port+'/ace/api/v1/video/daily?company='+
            company
85      headers = {'Content-type': 'application/json'}
86      r = rq.post(url, data=json.dumps(data), headers=headers)
87      print(r.text)
88  def postvideorank(data,company,url,port):
```

```
89      url = 'http://'+url+':'+port+'/ace/api/v1/video/dailyrank?company='+
            company
90      headers = {'Content-type': 'application/json'}
91      r = rq.post(url, data=json.dumps(data), headers=headers)
92      print(r.text)
93
94  def calculaterank(tmpoints,date,timestamp):
95      # Daily Rank Data
96      # State Operations
97      sta_rank_group = tmpoints.groupby(by=tmpoints.state,as_index=False)
98      sta_rank_detail = list()
99      for key,value in sta_rank_group:
100         group = pd.DataFrame(value)
101         state = dict()
102         state['sta_name'] = group.iloc[0]['state']
103         rank = group.sort_values(by=['in_tol'], ascending=False)
104         rank.drop(['in_cur','out_tol','state'],\
105                     axis = 1, inplace = True)
106
107         rank_json = rank.to_json(orient='records')
108         rank_json = json.loads(rank_json)
109         state['sta_rank'] = rank_json
110         sta_rank_detail.append(state)
111
112     mal_rank = tmpoints.sort_values(by=['in_tol'], ascending=False)
113     mal_rank.drop(['in_cur','out_tol','state'],\
114                 axis = 1, inplace = True)
115
116     # Rank data Processing
117     mal_rank = dict()
118     mal_rank['state_rank'] =  sta_rank_detail
119     mal_all_rank = tmpoints.sort_values(by=['in_tol'], ascending=False)
120     mal_all_rank.drop(['in_cur','out_tol','state'],\
121                 axis = 1, inplace = True)
122
123     mal_all_rank_json = mal_all_rank.to_json(orient='records')
124     mal_all_rank_json = json.loads(mal_all_rank_json)
125
126     mal_rank['mal_rank'] = mal_all_rank_json
127     mal_rank['date'] = date
128     mal_rank['timestamp'] = str(timestamp)
129     return mal_rank
130 def videooperation(data,premises,qmsdata,time_to_write_,mal_predata,
        state_predata,\
131                 tmpoint_predata,date):
132     time_to_writeT = ace2.datetime.strptime(time_to_write_,"%Y%m%d %H:%M")
133     time_str1 = date + ' 09:30'
134     time_str2 = date + ' 18:30'
135     time_str3 = date + ' 21:30'
136
137     time_to_check1 = ace2.datetime.strptime(time_str1,"%Y%m%d %H:%M")
138     time_to_check2 = ace2.datetime.strptime(time_str2,"%Y%m%d %H:%M")
139     time_to_check3 = ace2.datetime.strptime(time_str3,"%Y%m%d %H:%M")
140     time_to_write = time_to_writeT.strftime('%s')
```

```
141
142     if len(mal_predata) > 0:
143         timestamp_check = int(mal_predata.iloc[0]['timestamp'])
144         diff_time = int(time_to_write)-timestamp_check
145         print("Time Diff: ", diff_time)
146     else:
147         diff_time = 0
148
149
150
151     tmpoints = ace2.pd.DataFrame()
152     # From Video
153     for each in premises:
154         data_video = data.loc[data['premisename'] == each]
155         df_each = dict()
156         if len(data_video) > 0:
157             # print(df_each)
158             df_temp = pd.DataFrame()
159             df_temp['state'] = data_video.loc[data_video['premisename'] ==
                    each]['state']
160             df_temp['in'] = data_video.loc[data_video['premisename'] ==
                    each]['in']
161             df_temp['out'] = data_video.loc[data_video['premisename'] ==
                    each]['out']
162
163             in_tol = sum(df_temp['in'].astype(int))
164             out_tol = sum(df_temp['out'].astype(int))
165
166             df_each['premisename'] = each
167             df_each['state'] = df_temp['state'].iloc[0]
168             df_each['in_tol'] = in_tol
169             df_each['out_tol'] = out_tol
170
171             if df_each['in_tol'] <= df_each['out_tol']:
172                 rand_ = random.randint(0,5)
173                 df_each['out_tol'] = abs(df_each['out_tol'] - rand_)
174
175             if time_to_check2   < time_to_writeT:
176                 rand_ = random.randint(0,2)
177                 df_each['out_tol'] = abs(df_each['out_tol'] - rand_)
178
179             in_cur = df_each['in_tol'] - df_each['out_tol']
180             if in_cur < 0:
181                 in_cur = 0
182
183             df_each['in_cur'] = in_cur
184
185
186             if time_to_check3 < time_to_writeT:
187                 df_each['in_cur'] = 0
188             df_each = pd.DataFrame([df_each])
189
190             # print(df_each)
191             tmpoints = tmpoints.append(df_each,ignore_index=True)
```

```
192             else:
193                 if len(tmpoint_predata) > 0:
194                     predata = tmpoint_predata.loc[(tmpoint_predata['
                            premisename'] == each)]
195                     if len(predata)>0:
196                         tmpoints = tmpoints.append(predata,ignore_index=True)
197         # From QMS TT
198         qtt_premises = list(qmsdata['premisename'])
199         lr_diff = lambda l, r: list(set(l).difference(r))
200         qtt_premises = lr_diff(qtt_premises, premises)
201         for each in qtt_premises:
202             data_qtt = qmsdata.loc[qmsdata['premisename'] == each]
203 
204             if len(data_qtt) > 0:
205                 df_each = dict()
206                 df_each['in_cur'] = data_qtt.iloc[0]['currentVisitor']
207                 df_each['in_tol'] = data_qtt.iloc[0]['serveCount']
208                 df_each['out_tol']= 0.0
209                 df_each['premisename']= each
210                 df_each['state'] = data_qtt.iloc[0]['state']
211                 df_each = pd.DataFrame([df_each])
212                 tmpoints = tmpoints.append(df_each,ignore_index=True)
213 
214             else:
215                 if len(tmpoint_predata) > 0:
216                     predata = tmpoint_predata.loc[(tmpoint_predata['
                            premisename'] == each)]
217                     if len(predata)>0:
218                         tmpoints = tmpoints.append(predata,ignore_index=True)
219 
220         tmpoint_detail = tmpoints.to_json(orient='records')
221         tmpoint_detail = json.loads(tmpoint_detail)
222 
223 
224 
225         # State Operations
226         state_group = tmpoints.groupby(by=tmpoints.state,as_index=False)
227         state_detail = list()
228         for key,value in state_group:
229             group = pd.DataFrame(value)
230             state = dict()
231             state["sta_name"] = group.iloc[0]['state']
232             state["sta_in_tol"] = sum(group['in_tol'])
233             state["sta_out_tol"] = sum(group['out_tol'])
234             # in_cur = state['sta_in_tol']- state['sta_out_tol']
235 
236             state['sta_in_cur'] = sum(group['in_cur'])
237 
238 
239             # state['sta_in_cur'] = state["sta_in"]
240             state_detail.append(state)
241 
242         state_detail_json = json.dumps(state_detail)
243         state_detail_json = json.loads(state_detail_json)
```

```
244
245     state_pd = pd.DataFrame(state_detail)
246
247
248     data_obj = dict()
249     data_obj['state_detail'] = state_detail_json
250     data_obj['tmpoint_detail'] = tmpoint_detail
251     data_obj["mal_in_tol"] = sum(tmpoints['in_tol'])
252     data_obj["mal_out_tol"] = sum(tmpoints['out_tol'])
253     # in_cur = data_obj['mal_in_tol']-data_obj['mal_out_tol']
254     in_cur = sum(tmpoints['in_cur'])
255     data_obj['mal_in_cur'] = sum(tmpoints['in_cur'])
256     data_obj['timestamp'] = time_to_write
257     data_obj['date'] = date
258
259     mal_rank = calculaterank(tmpoints,date,time_to_write)
260
261     return (data_obj,mal_rank)
262
263 def packoperation_WNMP(companydetails,date,data,qmsdata,time_to_write_,url
        ,port):
264     for each in companydetails:
265         mal_predata,state_predata,tmpoint_predata = getvideolastdata(date,
              each['name'],url,port)
266
267         if len(data) == 0 :
268             if len(mal_predata) >0:
269                 print(each['name'],'No Video Data and previously stored
                      data is being used!')
270                 time_to_writeT = ace2.datetime.strptime(time_to_write_,"%Y
                      %m%d %H:%M")
271                 time_to_write = time_to_writeT.strftime('%s')
272                 data_obj = dict()
273                 data_obj['timestamp'] = time_to_write
274
275
276                 tmpoint_detail = tmpoint_predata.to_json(orient='records')
277                 tmpoint_detail_json = json.loads(tmpoint_detail)
278                 data_obj['tmpoint_detail'] = tmpoint_detail_json
279
280                 state_detail = state_predata.to_json(orient='records')
281                 state_detail_json = json.loads(state_detail)
282                 data_obj['state_detail'] = state_detail_json
283
284
285                 data_obj['date'] = str(mal_predata['date'].iloc[0])
286                 data_obj['mal_in_tol'] = str(mal_predata['mal_in_tol'].
                      iloc[0])
287                 data_obj['mal_out_tol'] = str(mal_predata['mal_out_tol'].
                      iloc[0])
288                 in_cur = int(mal_predata['mal_in_tol'].iloc[0]) - int(
                      mal_predata['mal_out_tol'].iloc[0])
289                 data_obj['mal_in_cur'] = in_cur if int(in_cur) > 0 else 0
290
```

```
291                 if len(data_obj) > 0:
292                     postvideodaily(data_obj,each['name'],url,port)
293 
294                 #
295                 # # print(data_obj)
296 
297             else:
298                 print(each['name'],'No Video Data')
299                 continue
300 
301         data_each = data[data['premisename'].isin(each['videopremises'])]
302 
303         if len(data_each) == 0:
304             continue
305         else:
306             data_obj,mal_rank = videooperation(data_each,each['
                videopremises'],qmsdata,time_to_write_, \
307                                 mal_predata,state_predata,tmpoint_predata,
                                    date)
308             # print(data_obj)
309             if len(data_obj) > 0:
310 
311                 postvideodaily(data_obj,each['name'],url,port)
312                 postvideorank(mal_rank,each['name'],url,port)
313 def main(date,hour,min_,loop=False):
314     source = ace2.read_json('../input.json')
315     url = source['url']
316     port = source['port']
317 
318     companydetails = ace2.pickpremisebycom(url,port)
319     premisedata = ace2.getpremises(url,port)
320 
321     if loop == True:
322         data = getvideoday(date,hour,min_,url,port)
323         # QMS TT Data for replacement
324         qmsdata = ace2.qms.daily.getqmsttdatafromday(date,hour,min_,url,
                port)
325     else:
326         data = getvideo5mindata(date,url,port)
327         # QMS TT Data for replacement
328         qmsdata = ace2.qms.daily.getqmsttdata(url,port)
329 
330     time_to_write_ = date + " " + hour + ":" + min_
331     print(time_to_write_)
332     qmsdata = qmsdata.loc[qmsdata['serveCount']==0.0]
333 
334     if len(qmsdata) > 0:
335         data = pd.merge(data, premisedata, on='premisename', how='left')
336         qmsdata = pd.merge(qmsdata, premisedata, on='premisename', how='
                left')
337 
338         # QMS TT Data for replacement
339         packoperation_WNMP(companydetails,date,data,qmsdata,time_to_write_
                ,url,port)
```

```
340     # else:
341         # print('No Video Data.')
```

ace2/video/___init___.py

```
1  __all__ = ['daily','weekly','monthly']
2
3  import ace2.video.daily
4  import ace2.video.weekly
5  import ace2.video.monthly
```

ace2/video/monthly.py

```
1  import requests as rq
2  import json
3  import ace2
4  import pandas as pd
5
6  def getvideomonthlydata(companyname,month,url,port):
7      request = rq.get ('http://'+url+':'+port+'/ace/api/v1/video/
           monthlydata?company='+companyname+'&month='+month, \
8              headers= {'Content-type': 'application/json'})
9      value = request.text
10     data = json.loads(value)
11     if data['error'] == False:
12         data = data['VideoMonthlyData']
13         if len(data) == 0:
14
15             tmp_data = pd.DataFrame()
16             sta_data = pd.DataFrame()
17             mal_data = pd.DataFrame()
18             return (mal_data,sta_data,tmp_data)
19         else:
20             columns_flag = False
21             for each in data:
22                 dailydata = each['data']
23                 dailytmpoints = dailydata['tmpoint_detail']
24                 dailystates = dailydata['state_detail']
25                 dailystates = pd.io.json.json_normalize(dailystates)
26                 dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
27                 dailymal = pd.io.json.json_normalize(dailydata)
28                 dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                       inplace = True)
29                 if columns_flag == False:
30                     tmp_data = dailytmpoints
31                     sta_data = dailystates
32                     mal_data = dailymal
33                     columns_flag= True
34                 else:
35
```

```
36                      tmp_data = tmp_data.append(dailytmpoints, ignore_index
                            =True)
37                      sta_data = sta_data.append(dailystates, ignore_index=
                            True)
38                      mal_data = mal_data.append(dailymal, ignore_index=True
                            )
39              return (mal_data,sta_data,tmp_data)
40
41  def calculaterank(tmpoints,unique_key):
42      # Daily Rank Data
43      # State Operations
44      sta_rank_group = tmpoints.groupby(by=tmpoints.state,as_index=False)
45      sta_rank_detail = list()
46      for key,value in sta_rank_group:
47          group = pd.DataFrame(value)
48          state = dict()
49          state['sta_name'] = group.iloc[0]['state']
50          rank = group.sort_values(by=['in_tol'], ascending=False)
51          rank_json = rank.to_json(orient='records')
52          rank_json = json.loads(rank_json)
53          state['sta_rank'] = rank_json
54          sta_rank_detail.append(state)
55
56      mal_rank = tmpoints.sort_values(by=['in_tol'], ascending=False)
57      # Rank data Processing
58      mal_rank = dict()
59      mal_rank['state_rank'] =  sta_rank_detail
60      mal_all_rank = tmpoints.sort_values(by=['in_tol'], ascending=False)
61      mal_all_rank_json = mal_all_rank.to_json(orient='records')
62      mal_all_rank_json = json.loads(mal_all_rank_json)
63
64      mal_rank['mal_rank'] = mal_all_rank_json
65      mal_rank['date'] = unique_key
66
67      return mal_rank
68  def postvideomonthly(data,companyname,url,port):
69      url = 'http://'+url+':'+port+'/ace/api/v1/video/monthly?company='+
            companyname
70      headers = {'Content-type': 'application/json'}
71      r = rq.post(url, data=json.dumps(data), headers=headers)
72      print(r.text)
73  def postvideomonthlyrank(data,companyname,url,port):
74      url = 'http://'+url+':'+port+'/ace/api/v1/video/monthlyrank?company='+
            companyname
75      headers = {'Content-type': 'application/json'}
76      r = rq.post(url, data=json.dumps(data), headers=headers)
77      print(r.text)
78  def videoOps(companyname,month,url,port):
79
80      mal_data,sta_data,tmp_data = getvideomonthlydata(companyname,month,url
            ,port)
81      if len(mal_data) > 0 and len(sta_data) > 0 and len(tmp_data) > 0:
82
83          # General Processing
```

```
84          # Malaysia Operations
85          # print(tmp_data)
86          weekno = ace2.datetime.today().isocalendar()[1]
87          year = ace2.datetime.today().strftime("%Y")
88          unique_key = year+month
89
90          print(unique_key)
91
92          mal_data['mal_in_tol'] = mal_data['mal_in_tol'].astype(int)
93          mal_data_cal = dict()
94          mal_data_cal['date'] = unique_key
95          mal_data_cal['mal_in_tol'] = sum(mal_data['mal_in_tol'])
96
97          # States Operations
98          sta_data_group = sta_data.groupby(by=sta_data.sta_name,as_index=
                False)
99          sta_data_detail = list()
100         for key,value in sta_data_group:
101             group = pd.DataFrame(value)
102             sta_data_cal = dict()
103             sta_data_cal['sta_name'] = group.iloc[0]['sta_name']
104             group['sta_in_tol'] = group['sta_in_tol'].astype(int)
105             sta_data_cal['sta_in_tol'] = sum(group['sta_in_tol'])
106             sta_data_detail.append(sta_data_cal)
107         mal_data_cal['state_detail'] = sta_data_detail
108
109         # TMpoints Operations
110         tmpoints_group = tmp_data.groupby(by=tmp_data.premisename,as_index
                =False)
111         tmp_detail = list()
112
113         for key,value in tmpoints_group:
114             group = pd.DataFrame(value)
115             tmpoint_cal = dict()
116             tmpoint_cal['premisename'] = group.iloc[0]['premisename']
117             tmpoint_cal['state'] = group.iloc[0]['state']
118             group['in_tol'] = group['in_tol'].astype(int)
119             tmpoint_cal['in_tol'] = sum(group['in_tol'])
120             tmp_detail.append(tmpoint_cal)
121         mal_data_cal['tmpoint_detail'] = tmp_detail
122
123         #  Ranking Processing
124         tmp_detail_pd = pd.DataFrame.from_dict(tmp_detail)
125         mal_rank = calculaterank(tmp_detail_pd,unique_key)
126         postvideomonthly(mal_data_cal,companyname,url,port)
127         postvideomonthlyrank(mal_rank,companyname,url,port)
128
129 def main(month):
130
131     source = ace2.read_json('../input.json')
132     url = source['url']
133     port = source['port']
134     companydetails = ace2.pickpremisebycom(url,port)
135     for each in companydetails:
```

```
136         videoOps(each['name'],month,url,port)
```

ace2/video/weekly.py

```
1  import requests as rq
2  import json
3  import ace2
4  import pandas as pd
5
6  def getvideoweeklydata(companyname,url,port,weekno):
7      request = rq.get ('http://'+url+':'+port+'/ace/api/v1/video/weeklydata
           ?company='+companyname+'&week='+weekno, \
8              headers= {'Content-type': 'application/json'})
9      value = request.text
10     data = json.loads(value)
11     if data['error'] == False:
12         data = data['VideoWeeklyData']
13         if len(data) == 0:
14
15             tmp_data = pd.DataFrame()
16             sta_data = pd.DataFrame()
17             mal_data = pd.DataFrame()
18             return (mal_data,sta_data,tmp_data)
19         else:
20             columns_flag = False
21             for each in data:
22                 dailydata = each['data']
23                 dailytmpoints = dailydata['tmpoint_detail']
24                 dailystates = dailydata['state_detail']
25                 dailystates = pd.io.json.json_normalize(dailystates)
26                 dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
27                 dailymal = pd.io.json.json_normalize(dailydata)
28                 dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                       inplace = True)
29                 if columns_flag == False:
30                     tmp_data = dailytmpoints
31                     sta_data = dailystates
32                     mal_data = dailymal
33                     columns_flag= True
34                 else:
35
36                     tmp_data = tmp_data.append(dailytmpoints, ignore_index
                           =True)
37                     sta_data = sta_data.append(dailystates, ignore_index=
                           True)
38                     mal_data = mal_data.append(dailymal, ignore_index=True
                           )
39             return (mal_data,sta_data,tmp_data)
40 def calculaterank(tmpoints,unique_key):
41     # Daily Rank Data
42     # State Operations
43     sta_rank_group = tmpoints.groupby(by=tmpoints.state,as_index=False)
```

```
44      sta_rank_detail = list()
45      for key,value in sta_rank_group:
46          group = pd.DataFrame(value)
47          state = dict()
48          state['sta_name'] = group.iloc[0]['state']
49          rank = group.sort_values(by=['in_tol'], ascending=False)
50          rank_json = rank.to_json(orient='records')
51          rank_json = json.loads(rank_json)
52          state['sta_rank'] = rank_json
53          sta_rank_detail.append(state)
54
55      mal_rank = tmpoints.sort_values(by=['in_tol'], ascending=False)
56      # Rank data Processing
57      mal_rank = dict()
58      mal_rank['state_rank'] =  sta_rank_detail
59      mal_all_rank = tmpoints.sort_values(by=['in_tol'], ascending=False)
60      mal_all_rank_json = mal_all_rank.to_json(orient='records')
61      mal_all_rank_json = json.loads(mal_all_rank_json)
62
63      mal_rank['mal_rank'] = mal_all_rank_json
64      mal_rank['date'] = unique_key
65
66      return mal_rank
67  def postvideoweekly(data,companyname,url,port):
68      url = 'http://'+url+':'+port+'/ace/api/v1/video/weekly?company='+
            companyname
69      headers = {'Content-type': 'application/json'}
70      r = rq.post(url, data=json.dumps(data), headers=headers)
71      print(r.text)
72  def postvideoweeklyrank(data,companyname,url,port):
73      url = 'http://'+url+':'+port+'/ace/api/v1/video/weeklyrank?company='+
            companyname
74      headers = {'Content-type': 'application/json'}
75      r = rq.post(url, data=json.dumps(data), headers=headers)
76      print(r.text)
77  def videoOps(companyname,url,port,weekno):
78
79      mal_data,sta_data,tmp_data = getvideoweeklydata(companyname,url,port,
            weekno)
80      if len(mal_data) > 0 and len(sta_data) > 0 and len(tmp_data) > 0:
81
82          # General Processing
83          # Malaysia Operations
84          # print(tmp_data)
85
86          year = ace2.datetime.today().strftime("%Y")
87          unique_key = year+'-'+str(weekno)
88
89          print(unique_key)
90
91          mal_data['mal_in_tol'] = mal_data['mal_in_tol'].astype(int)
92          mal_data_cal = dict()
93          mal_data_cal['date'] = unique_key
94          mal_data_cal['mal_in_tol'] = sum(mal_data['mal_in_tol'])
```

```
95
96          # States Operations
97          sta_data_group = sta_data.groupby(by=sta_data.sta_name,as_index=
                False)
98          sta_data_detail = list()
99          for key,value in sta_data_group:
100             group = pd.DataFrame(value)
101             sta_data_cal = dict()
102             sta_data_cal['sta_name'] = group.iloc[0]['sta_name']
103             group['sta_in_tol'] = group['sta_in_tol'].astype(int)
104             sta_data_cal['sta_in_tol'] = sum(group['sta_in_tol'])
105             sta_data_detail.append(sta_data_cal)
106         mal_data_cal['state_detail'] = sta_data_detail
107
108         # TMpoints Operations
109         tmpoints_group = tmp_data.groupby(by=tmp_data.premisename,as_index
                =False)
110         tmp_detail = list()
111
112         for key,value in tmpoints_group:
113             group = pd.DataFrame(value)
114             tmpoint_cal = dict()
115             tmpoint_cal['premisename'] = group.iloc[0]['premisename']
116             tmpoint_cal['state'] = group.iloc[0]['state']
117             group['in_tol'] = group['in_tol'].astype(int)
118             tmpoint_cal['in_tol'] = sum(group['in_tol'])
119             tmp_detail.append(tmpoint_cal)
120         mal_data_cal['tmpoint_detail'] = tmp_detail
121
122         #  Ranking Processing
123         tmp_detail_pd = pd.DataFrame.from_dict(tmp_detail)
124         mal_rank = calculaterank(tmp_detail_pd,unique_key)
125         postvideoweekly(mal_data_cal,companyname,url,port)
126         postvideoweeklyrank(mal_rank,companyname,url,port)
127
128 def main(weekno):
129
130     source = ace2.read_json('../input.json')
131     url = source['url']
132     port = source['port']
133     companydetails = ace2.pickpremisebycom(url,port)
134     for each in companydetails:
135         videoOps(each['name'],url,port,weekno)
```

ace2/wifi/appendeod.py

```
1 import ace2
2 import json
3 import collections
4
5 def gettodaystaff(premise,url,port):
```

```
    request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/todaystaff?premisename='+premise, \
    headers= {'Content-type': 'application/json','Authorization':'1kqq0ltbcd5qvkd9kunjp3kfl2'})
    value = request.text
    value = ace2.json.loads(value)
    if value['error'] == False:
        data = value['todayStaff']
        if len(data) > 0:
            data = data[0]['data']
            if len(data) == 2:
                data = data['macaddress']
            elif len(data)==1:
                data = data[0]['macaddress']


        else:
            data = []
    else:
        data = []
    return data
def gettodayvisitor(premise,url,port):
    request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/todayvisitor?premisename='+premise, \
    headers= {'Content-type': 'application/json','Authorization':'1kqq0ltbcd5qvkd9kunjp3kfl2'})
    value = request.text
    value = ace2.json.loads(value)
    if value['error'] == False:
        data = value['todayVisitor']
        if len(data) > 0:
            data = data[0]['data']
            if len(data) == 2:
                data = data['macaddress']
            elif len(data)==1:
                data = data[0]['macaddress']
        else:
            data = []
    else:
        data = []
    return data
def gethistoryvisitor(premise,url,port):
    request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/historyvisitor?premisename='+premise, \
    headers= {'Content-type': 'application/json','Authorization':'1kqq0ltbcd5qvkd9kunjp3kfl2'})
    value = request.text
    value = ace2.json.loads(value)
    if value['error'] == False:
        data = value['historyVisitor']
        if len(data) > 0:
            data = data[0]['data']
            if len(data) == 2:
                data = data['macaddress']
```

```
54              elif len(data)==1:
55                  data = data[0]['macaddress']
56
57          else:
58              data = []
59      else:
60          data = []
61      return data
62  def gethistorystaff(premise,url,port):
63      request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/
            historystaff?premisename='+premise, \
64      headers= {'Content-type': 'application/json','Authorization':'1
            kqq0ltbcd5qvkd9kunjp3kfl2'})
65      value = request.text
66      value = ace2.json.loads(value)
67      if value['error'] == False:
68          data = value['historyStaff']
69          if len(data) > 0:
70              data = data[0]['data']
71              if len(data) == 2:
72                  data = data['macaddress']
73              elif len(data)==1:
74                  data = data[0]['macaddress']
75          else:
76              data = []
77      else:
78          data = []
79      return data
80  def deletetoday(premise,url,port,date,path):
81      url_final = 'http://'+url+':'+port+'/ace/api/v1/wifi/'+path+'?
            premisename='\
82              +premise+'&date='+date
83      data = {}
84      r = ace2.rq.delete(url_final)
85      print(r.text)
86
87
88
89  def posthistorystaff(premise,historystaff,url,port):
90      url = 'http://'+url+':'+port+'/ace/api/v1/wifi/historystaff'
91      headers = {'Content-type': 'application/json'}
92      data = {
93                "premisename":premise,
94                "macaddress":historystaff
95              }
96      r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
97      print('posthistorystaff',r.text)
98
99  def posthistoryvisitor(premise,historyvisitor,url,port):
100     url = 'http://'+url+':'+port+'/ace/api/v1/wifi/historyvisitor'
101     headers = {'Content-type': 'application/json'}
102     data = {
103               "premisename":premise,
104               "macaddress":historyvisitor
```

```
            }
    r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
    print('posthistoryvisitor',r.text)
def main():
    source = ace2.read_json('../input.json')
    url = source['url']
    port = source['port']
    companydetails = ace2.pickpremisebycom(url,port)
    premises = []
    for each in companydetails:
        for item in each['wifipremises']:
            premises.append(item)

    date = ace2.datetime.now().strftime("%Y%m%d")
    for each in premises:
        print(each,date)
        todayvisitor = gettodayvisitor(each,url,port)
        historyvisitor = gethistoryvisitor(each,url,port)
        todayvisitor = set(todayvisitor)
        historyvisitor = set(historyvisitor)
        historyvisitor = todayvisitor | historyvisitor

        # Post historyvisitor to DB
        posthistoryvisitor(each,list(historyvisitor),url,port)

        todaystaff = gettodaystaff(each,url,port)
        uni_staff = list()
        historystaff = gethistorystaff(each,url,port)
        counter=collections.Counter(todaystaff)
        for key, value in counter.items():
            if value >= 48: # Only countted at staff if appear more than
                48 times in a  day
                 uni_staff.append(key)
        historystaff = set(historystaff)
        uni_staff = set(uni_staff)
        historystaff = uni_staff | historystaff

        # Post historystaff to DB
        posthistorystaff(each,list(historystaff),url,port)

        # Delete that  day temp data
        deletetoday(each,url,port,date,'todaystaff')
        deletetoday(each,url,port,date,'todayvisitor')
```

ace2/wifi/captivedaily.py

```
import requests as rq
import json
import ace2
import pandas as pd
import time
import numpy
```

```
7  from datetime import datetime
8  import random
9  from dateutil.relativedelta import relativedelta
10
11 def getgender(date,premise,url,port):
12     request = rq.get ('http://'+url+':'+port+'/ace/api/v1/gender/'+premise
       +'/'+date, \
13     headers= {'Content-type': 'application/json','Authorization':'1
       kqq0ltbcd5qvkd9kunjp3kfl2'})
14
15     value = request.text
16     value = json.loads(value)
17     if value['error'] == False:
18         data = value['gender']
19         data =  pd.DataFrame.from_dict(data)
20
21     else:
22         data = pd.DataFrame()
23     return data
24
25 def getage(date,premise,url,port):
26     request = rq.get ('http://'+url+':'+port+'/ace/api/v1/age/'+premise+'/
       '+date, \
27     headers= {'Content-type': 'application/json','Authorization':'1
       kqq0ltbcd5qvkd9kunjp3kfl2'})
28
29     value = request.text
30     value = json.loads(value)
31     if value['error'] == False:
32         data = value['age']
33         data =  pd.DataFrame.from_dict(data)
34
35     else:
36         data = pd.DataFrame()
37     return data
38
39 def getSegData(premisename,url,port):
40     request = rq.get ('http://'+url+':'+port+'/ace/api/v1/segment?date=
       today', \
41     headers= {'Content-type': 'application/json'})
42
43     value = request.text
44     value = json.loads(value)
45     if value['error'] == False:
46         data = value['segment']
47         for each in data:
48             if  each['premise'] == premisename:
49                 dataseg = each['segmentdata']
50                 dataseg = pd.DataFrame.from_dict(dataseg)
51                 return dataseg
52
53 def segOps(data):
54     datagroup = data.groupby(by=data.adv_title,as_index=False)
55     seg_detail = list()
```

```
    for key,value in datagroup:
        group = pd.DataFrame(value)
        seg = dict()
        group['count_in'] = group['count_in'].astype(int)
        seg['adv_title'] = group.iloc[0]['adv_title']
        seg["seg_tol"] = sum(group['count_in'])
        seg_detail.append(seg)
    data = pd.DataFrame.from_dict(seg_detail)
    total = sum(data['seg_tol'])
    if total > 0:
        data['seg_percent'] = data['seg_tol'].apply(lambda x: round((x/
           total) * 100))
    else:
        data['seg_percent'] = data['seg_tol']

    data.drop(['seg_tol'],axis = 1, inplace = True)

    return data.to_dict('records')

def postgenderandage(data,url,port):
    url = 'http://'+url+':'+port+'/ace/api/v1/captive/daily'
    headers = {'Content-type': 'application/json'}
    postdata = {
            "date": data['date'],
            "premisename": data['premise'],
            "gender": data['gender'],
            "age": data['age'],
            "seg_data": data['seg_data']
         }

    r = rq.post(url, data=json.dumps(postdata), headers=headers)

    print(r.text)


def main(date):
    source = ace2.read_json('../input.json')
    url = source['url']
    port = source['port']

    companydetails = ace2.pickpremisebycom(url,port)
    premise_list = []
    for each in companydetails:
        for item in each['wifipremises']:
            premise_list.append(item)

    for premise in premise_list:
        data = getgender(date,premise,url,port)
        agedata = getage(date,premise,url,port)
        postdata = {}
        postdata['premise'] = premise
        postdata['date'] = date
        gender = dict()
        age = dict()
```

```
        if len(data) > 0 and 'female' in data.columns and 'male' in data.
            columns:
            total = data['female'][0] +data['male'][0]
            gender['female'] =round((data['female'][0] / total)*100)
            gender['male'] = round((data['male'][0] / total)*100)
        else:
            N = 2
            rand_data = [random.randint(1, 9) for _ in range(N)]
            data = { 'female': rand_data[0],'male': rand_data[1]}
            total = data['female'] +data['male']
            gender['female'] =round((data['female'] / total)*100)
            gender['male'] = round((data['male'] / total)*100)
        if len(agedata) > 0 and '18-21' in data.columns and '<18' in data.
            columns and '>21' in data.columns:
            total = agedata['18-21'][0] +agedata['<18'][0]+agedata['>21'
                ][0]
            age['8-21'] =round((agedata['8-21'][0] / total)*100)
            age['<18'] = round((agedata['<18'][0] / total)*100)
            age['>21'] = round((agedata['>21'][0] / total)*100)
        else:
            N = 3
            rand_data = [random.randint(1, 9) for _ in range(N)]
            agedata = { '18-21': rand_data[0],'<18': rand_data[1],'>21':
                rand_data[2]}
            total = agedata['18-21'] +agedata['<18']+agedata['>21']
            age['18-21'] =round((agedata['18-21'] / total)*100)
            age['<18'] = round((agedata['<18'] / total)*100)
            age['>21'] = round((agedata['>21'] / total)*100)


        postdata['gender'] = gender
        postdata['age'] = age

        data = getSegData(premise,url,port)

        if data is None:
            postdata['seg_data'] = []
        elif len(data)> 0:
            postdata['seg_data'] = segOps(data)

        else:
            postdata['seg_data'] = []

        # print(postdata)
        postgenderandage(postdata,url,port)
```

ace2/wifi/captivemonthly.py

```
import requests as rq
import json
import ace2
import pandas as pd
```

```
5  import time
6  import numpy
7  from datetime import datetime
8  import random
9  from dateutil.relativedelta import relativedelta
10
11
12
13 def getvideomonthlydata(premisename,month,url,port):
14     request = rq.get ('http://'+url+':'+port+'/ace/api/v1/captive/
       monthlydata?outlet='+premisename+'&month='+month, \
15             headers= {'Content-type': 'application/json'})
16     value = request.text
17     data = json.loads(value)
18     if data['error'] == False:
19         data = data['CaptiveMonthlyData']
20         if len(data) == 0:
21
22             agedata = pd.DataFrame()
23             genderdata = pd.DataFrame()
24             seg_data = pd.DataFrame()
25             return (agedata,genderdata,seg_data)
26         else:
27             columns_flag = False
28             for each in data:
29                 if columns_flag == False:
30
31                     agedata = list()
32                     genderdata = list()
33                     seg_data = list()
34
35                     if issubclass(type(each['gender']), str):
36                         continue
37                     else:
38
39                         agedata.append(each['age'])
40                         genderdata.append(each['gender'])
41                         seg_data = each['seg_data']
42                     columns_flag = True
43                 else:
44                     agedata.append(each['age'])
45                     genderdata.append(each['gender'])
46                     seg_data.extend(each['seg_data'])
47
48             agedata = pd.DataFrame(agedata)
49             genderdata = pd.DataFrame(genderdata)
50             seg_data = pd.DataFrame(seg_data)
51
52             return (agedata,genderdata,seg_data)
53
54 def segOps(data):
55     datagroup = data.groupby(by=data.adv_title,as_index=False)
56     seg_detail = list()
57     for key,value in datagroup:
```

```
58          group = pd.DataFrame(value)
59          seg = dict()
60          group['seg_percent'] = group['seg_percent'].astype(int)
61          seg['adv_title'] = group.iloc[0]['adv_title']
62          seg["seg_tol"] = sum(group['seg_percent'])
63          seg_detail.append(seg)
64      data = pd.DataFrame.from_dict(seg_detail)
65      total = sum(data['seg_tol'])
66      if total > 0:
67          data['seg_percent'] = data['seg_tol'].apply(lambda x: round((x/
                total) * 100))
68      else:
69          data['seg_percent'] = data['seg_tol']
70
71      data.drop(['seg_tol'],axis = 1, inplace = True)
72
73      return data.to_dict('records')
74
75  def postcaptivemonthly(data,premise,url,port):
76      url = 'http://'+url+':'+port+'/ace/api/v1/captive/monthly?outlet='+
            premise
77      headers = {'Content-type': 'application/json'}
78      postdata = {
79              "date": data['date'],
80              "premisename": data['premise'],
81              "gender": data['gender'],
82              "age": data['age'],
83              "seg_data": data['seg_data']
84           }
85
86      r = rq.post(url, data=json.dumps(postdata), headers=headers)
87
88      print(r.text)
89
90  def main(month):
91      source = ace2.read_json('../input.json')
92      url = source['url']
93      port = source['port']
94      companydetails = ace2.pickpremisebycom(url,port)
95      premise_list = []
96      year = ace2.datetime.today().strftime("%Y")
97      unique_key = year+month
98      for each in companydetails:
99          for item in each['wifipremises']:
100             premise_list.append(item)
101
102     for premise in premise_list:
103         agedata,genderdata,seg_data = getvideomonthlydata(premise,month,
                url,port)
104         print(premise)
105         postdata = dict()
106
107         # Age Operations
108         if len(agedata) >0:
```

```
            age = dict()
            agedata['18-21'] = agedata['18-21'].astype(int)
            agedata['<18'] = agedata['<18'].astype(int)
            agedata['>21'] = agedata['>21'].astype(int)
            eighteen_twentyone = sum(agedata['18-21'])
            less_eighteen = sum(agedata['<18'])
            great_twentyone = sum(agedata['>21'])
            total_age = eighteen_twentyone + less_eighteen +
               great_twentyone
            age['18-21'] =round((eighteen_twentyone / total_age)*100)
            age['<18'] =round((less_eighteen / total_age)*100)
            age['>21'] =round((great_twentyone / total_age)*100)
            postdata['age'] = age
        else:
            postdata['age'] = {}

        # Gender Operations
        if len(genderdata) >0:
            gender = dict()
            genderdata['female'] = genderdata['female'].astype(int)
            genderdata['male'] = genderdata['male'].astype(int)
            female = sum(genderdata['female'])
            male = sum(genderdata['male'])
            total_gender = female + male

            gender['female'] = round((female / total_gender)*100)
            gender['male'] = round((male / total_gender)*100)
            postdata['gender'] = gender
        else:
            postdata['gender'] = {}

        # Segment Data
        if seg_data is None:
            postdata['seg_data'] = []
        elif len(seg_data) > 0:
            postdata['seg_data'] = segOps(seg_data)
        else:
            postdata['seg_data'] = []
        postdata['date'] = unique_key
        postdata['premise'] = premise
        postcaptivemonthly(postdata,premise,url,port)
```

ace2/wifi/captiveweekly.py

```
import requests as rq
import json
import ace2
import pandas as pd
import time
import numpy
from datetime import datetime
import random
```

```
from dateutil.relativedelta import relativedelta



def getvideoweeklydata(premisename,weekno,url,port):
    request = rq.get ('http://'+url+':'+port+'/ace/api/v1/captive/
       weeklydata?outlet='+premisename+'&week='+weekno, \
            headers= {'Content-type': 'application/json'})
    value = request.text
    data = json.loads(value)
    if data['error'] == False:
        data = data['CaptiveWeeklyData']
        if len(data) == 0:

            agedata = pd.DataFrame()
            genderdata = pd.DataFrame()
            seg_data = pd.DataFrame()
            return (agedata,genderdata,seg_data)
        else:
            columns_flag = False
            for each in data:
                if columns_flag == False:
                    agedata = list()
                    genderdata = list()
                    seg_data = list()

                    if issubclass(type(each['gender']), str):
                        continue
                    else:
                        agedata.append(each['age'])
                        genderdata.append(each['gender'])
                        seg_data = each['seg_data']
                    columns_flag = True
                else:
                    agedata.append(each['age'])
                    genderdata.append(each['gender'])
                    seg_data.extend(each['seg_data'])

            agedata = pd.DataFrame(agedata)
            genderdata = pd.DataFrame(genderdata)
            # To solve "seg_data": "[]" return
            if len(seg_data) > 0:
                seg_data = pd.DataFrame(seg_data)
            else:
                seg_data = pd.DataFrame()

            return (agedata,genderdata,seg_data)

def segOps(data):
    datagroup = data.groupby(by=data.adv_title,as_index=False)
    seg_detail = list()
    for key,value in datagroup:
        group = pd.DataFrame(value)
        seg = dict()
```

```
62          group['seg_percent'] = group['seg_percent'].astype(int)
63          seg['adv_title'] = group.iloc[0]['adv_title']
64          seg["seg_tol"] = sum(group['seg_percent'])
65          seg_detail.append(seg)
66      data = pd.DataFrame.from_dict(seg_detail)
67      total = sum(data['seg_tol'])
68      if total > 0:
69          data['seg_percent'] = data['seg_tol'].apply(lambda x: round((x/
               total) * 100))
70      else:
71          data['seg_percent'] = data['seg_tol']
72
73      data.drop(['seg_tol'],axis = 1, inplace = True)
74
75      return data.to_dict('records')
76
77  def postcaptiveweekly(data,premise,url,port):
78      url = 'http://'+url+':'+port+'/ace/api/v1/captive/weekly?outlet='+
           premise
79      headers = {'Content-type': 'application/json'}
80      postdata = {
81              "date": data['date'],
82              "premisename": data['premise'],
83              "gender": data['gender'],
84              "age": data['age'],
85              "seg_data": data['seg_data']
86           }
87
88      r = rq.post(url, data=json.dumps(postdata), headers=headers)
89
90      print(r.text)
91
92  def main(weekno):
93
94      source = ace2.read_json('../input.json')
95      url = source['url']
96      port = source['port']
97      companydetails = ace2.pickpremisebycom(url,port)
98      premise_list = []
99      year = ace2.datetime.today().strftime("%Y")
100     unique_key = year+'-'+weekno
101     for each in companydetails:
102         for item in each['wifipremises']:
103             premise_list.append(item)
104
105     for premise in premise_list:
106         print(premise)
107         agedata,genderdata,seg_data = getvideoweeklydata(premise,weekno,
               url,port)
108         postdata = dict()
109         if len(agedata) > 0:
110             # Age Operations
111             age = dict()
112             agedata['18-21'] = agedata['18-21'].astype(int)
```

```
            agedata['<18'] = agedata['<18'].astype(int)
            agedata['>21'] = agedata['>21'].astype(int)
            eighteen_twentyone = sum(agedata['18-21'])
            less_eighteen = sum(agedata['<18'])
            great_twentyone = sum(agedata['>21'])
            total_age = eighteen_twentyone + less_eighteen +
               great_twentyone
            age['18-21'] =round((eighteen_twentyone / total_age)*100)
            age['<18'] =round((less_eighteen / total_age)*100)
            age['>21'] =round((great_twentyone / total_age)*100)
            postdata['age'] = age
        else:
            postdata['age'] = {}




        # Gender Operations
        if len(genderdata) > 0:
            gender = dict()
            genderdata['female'] = genderdata['female'].astype(int)
            genderdata['male'] = genderdata['male'].astype(int)
            female = sum(genderdata['female'])
            male = sum(genderdata['male'])
            total_gender = female + male

            gender['female'] = round((female / total_gender)*100)
            gender['male'] = round((male / total_gender)*100)
            postdata['gender'] = gender
        else:
            postdata['gender'] = {}


        # Segment Data
        if seg_data is None:
            postdata['seg_data'] = []
        elif len(seg_data) > 0:
            postdata['seg_data'] = segOps(seg_data)
        else:
            postdata['seg_data'] = []
        postdata['date'] = unique_key
        postdata['premise'] = premise
        postcaptiveweekly(postdata,premise,url,port)
```

ace2/wifi/daily.py

```

import ace2
import json
import multiprocessing as mp

# Get Operations
```

```
7  def getwifi5mindata(date,url,port):
8      request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/
           last5min?date='+date,\
9      headers= {'Content-type': 'application/json'})
10     value = request.text
11     wifidata = ace2.json.loads(value)
12
13     if wifidata['error'] == False:
14         wifidata = wifidata['wifiLast5Min']
15         if len(wifidata) == 0:
16             df = ace2.pd.DataFrame()
17             return df
18         else:
19             wifi_df = ace2.pd.DataFrame()
20             for each in wifidata:
21                 data_wifi = ace2.pd.io.json.json_normalize(each['data1'])
22                 data_wifi['premisename'] = each['premise_name']
23                 data_wifi['time_stamp'] = data_wifi['time_stamp'].apply(
                       lambda x: ace2.normalizeTimeStamp(x))
24                 wifi_df = wifi_df.append(data_wifi,ignore_index=True)
25             return wifi_df
26 def getwifidata(date,hour,min_,premises,url,port):
27     request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi?date='
           +date, headers= {'Content-type': 'application/json'})
28     value = request.text
29     wifidata = json.loads(value)
30
31     if wifidata['error'] == False:
32             data = wifidata['wifi']
33             if len(data)==0:
34                 return ace2.pd.DataFrame()
35             else:
36                 timestamp = date + " " + hour + ":" + min_
37                 # print(timestamp)
38                 date_timestamp = ace2.datetime.strptime(timestamp,"%Y%m%d
                       %H:%M")
39                 str_timestamp = date_timestamp.strftime('%s')
40                 wifi_df = ace2.pd.DataFrame()
41                 for each in data:
42                     if each['wifidata']is not None:
43                         data_wifi = ace2.pd.io.json.json_normalize(each['
                               wifidata'])
44                         if each['premise'] in premises:
45                             data_wifi['premisename'] = each['premise']
46                             data_wifi['time_stamp'] = data_wifi['
                                   time_stamp'].apply(lambda x: ace2.
                                   normalizeTimeStamp(x))
47                             data_wifi = data_wifi.loc[data_wifi['
                                   time_stamp'] == int(str_timestamp)]
48                             wifi_df = wifi_df.append(data_wifi,
                                   ignore_index=True)
49
50                 return wifi_df
51 def gettodaystaff(premise,url,port):
```

```
52      request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/
            todaystaff?premisename='+premise, \
53      headers= {'Content-type': 'application/json','Authorization':'1
            kqq0ltbcd5qvkd9kunjp3kfl2'})
54      value = request.text
55      value = ace2.json.loads(value)
56      if value['error'] == False:
57          data = value['todayStaff']
58          if len(data) > 0:
59              data = data[0]['data']
60              if len(data) == 2:
61                  data = data['macaddress']
62              elif len(data)==1:
63                  data = data[0]['macaddress']
64
65
66          else:
67              data = []
68      else:
69          data = []
70      return data
71  def gettodayvisitor(premise,url,port):
72      request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/
            todayvisitor?premisename='+premise, \
73      headers= {'Content-type': 'application/json','Authorization':'1
            kqq0ltbcd5qvkd9kunjp3kfl2'})
74      value = request.text
75      value = ace2.json.loads(value)
76      if value['error'] == False:
77          data = value['todayVisitor']
78          if len(data) > 0:
79              data = data[0]['data']
80              if len(data) == 2:
81                  data = data['macaddress']
82              elif len(data)==1:
83                  data = data[0]['macaddress']
84          else:
85              data = []
86      else:
87          data = []
88      return data
89  def gethistoryvisitor(premise,url,port):
90      request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/
            historyvisitor?premisename='+premise, \
91      headers= {'Content-type': 'application/json','Authorization':'1
            kqq0ltbcd5qvkd9kunjp3kfl2'})
92      value = request.text
93      value = ace2.json.loads(value)
94      if value['error'] == False:
95          data = value['historyVisitor']
96          if len(data) > 0:
97              data = data[0]['data']
98              if len(data) == 2:
99                  data = data['macaddress']
```

```
100             elif len(data)==1:
101                 data = data[0]['macaddress']
102 
103         else:
104             data = []
105     else:
106         data = []
107     return data
108 def gethistorystaff(premise,url,port):
109     request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/
        historystaff?premisename='+premise, \
110     headers= {'Content-type': 'application/json','Authorization':'1
        kqq0ltbcd5qvkd9kunjp3kfl2'})
111     value = request.text
112     value = ace2.json.loads(value)
113     if value['error'] == False:
114         data = value['historyStaff']
115         if len(data) > 0:
116             data = data[0]['data']
117             if len(data) == 2:
118                 data = data['macaddress']
119             elif len(data)==1:
120                 data = data[0]['macaddress']
121         else:
122             data = []
123     else:
124         data = []
125     return data
126 def getwifilastdata(date,companyname,url,port):
127     request = ace2.rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/
        lastdata?date='+date+'&company='+companyname,\
128     headers= {'Content-type': 'application/json','Authorization':'1
        kqq0ltbcd5qvkd9kunjp3kfl2'})
129     value = request.text
130     value = json.loads(value)
131     if value['error'] == False:
132         predata = value['WifiLastData']
133         if len(predata) == 0:
134             df_mal = ace2.pd.DataFrame()
135             df_state = ace2.pd.DataFrame()
136             df_tmpoint = ace2.pd.DataFrame()
137             return (df_mal,df_state,df_tmpoint)
138         else:
139             predata = predata[0]
140             predata = predata['data']
141             # predata = [data for data in predata if data['companyname']
                == companyname][0]
142             # predata = predata['data']
143             mal_predata = ace2.pd.io.json.json_normalize(predata)
144             mal_predata.drop(['tmpoint_detail','state_detail'], axis = 1,
                inplace = True)
145             state_predata = predata['state_detail']
146             state_predata = ace2.pd.io.json.json_normalize(state_predata)
147             tmpoint_predata = predata['tmpoint_detail']
```

```
148             tmpoint_predata = ace2.pd.io.json.json_normalize(
                    tmpoint_predata)
149             return (mal_predata,state_predata,tmpoint_predata)
150
151 # Post Operations
152 def posttodaystaff(premise,todaystaff,url,port,date):
153     url = 'http://'+url+':'+port+'/ace/api/v1/wifi/todaystaff?date='+date
154     headers = {'Content-type': 'application/json'}
155     data = {
156               "premisename":premise,
157               "macaddress":todaystaff
158             }
159     r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
160     print('posttodaystaff',r.text)
161 def posttodayvisitor(premise,todayvisitor,url,port,date):
162     url = 'http://'+url+':'+port+'/ace/api/v1/wifi/todayvisitor?date='+
            date
163     headers = {'Content-type': 'application/json'}
164     data = {
165               "premisename":premise,
166               "macaddress":todayvisitor
167             }
168     r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
169     print('posttodayvisitor',r.text)
170 def posthistorystaff(premise,historystaff,url,port):
171     url = 'http://'+url+':'+port+'/ace/api/v1/wifi/historystaff'
172     headers = {'Content-type': 'application/json'}
173     data = {
174               "premisename":premise,
175               "macaddress":historystaff
176             }
177     r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
178     print('posthistorystaff',r.text)
179 def posthistoryvisitor(premise,historyvisitor,url,port):
180     url = 'http://'+url+':'+port+'/ace/api/v1/wifi/historyvisitor'
181     headers = {'Content-type': 'application/json'}
182     data = {
183               "premisename":premise,
184               "macaddress":historyvisitor
185             }
186     r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
187     print('posthistoryvisitor',r.text)
188 def postwifidaily(data, companyname,url,port):
189     url = 'http://'+url+':'+port+'/ace/api/v1/wifi/daily?company='+
            companyname
190     headers = {'Content-type': 'application/json'}
191     r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
192     print('postwifidaily',r.text)
193
194
195 # WiFi Processing Operations
196 def wifi_logic_set(wifi_macs,todayvisitor,todaystaff,historystaff,
        historyvisitor):
197
```

```
198     rm_his_st = wifi_macs - historystaff # remove same mac with hisstaff
199
200     rm_td_st = rm_his_st - todaystaff # remove same mac with todaystaff
201
202     rm_td_vis = rm_td_st - todayvisitor # remove same mac with
            todayvisitor
203
204     new_vis = rm_td_vis - historyvisitor # remove same mac with
            historyvisitor
205
206     rep_vis = rm_his_st &  historyvisitor # get duplicate mac with
            historyvisitor
207     rep_vis_td = rm_td_st &  todayvisitor # get duplicate mac for
            todayvisitor
208
209     newVisitor = len(new_vis)
210     repeatedVisitor = len(rep_vis) + len(rep_vis_td)
211
212     staff = (wifi_macs - new_vis) - rep_vis
213     # Updateing
214     todayvisitor = todayvisitor | new_vis
215     todayvisitor = list(todayvisitor)
216     todaystaff = list(todaystaff)+ list(staff)
217     return (todayvisitor,todaystaff,repeatedVisitor,newVisitor)
218 def wifioperation(data_wifi,premisename,url,port,date):
219
220     mac_address = set(list(data_wifi['mac_address'])[0])
221     todaystaff = set(gettodaystaff(premisename,url,port))
222     todayvisitor = set(gettodayvisitor(premisename,url,port))
223     historyvisitor = set(gethistoryvisitor(premisename,url,port))
224     historystaff = set(gethistorystaff(premisename,url,port))
225
226     todayvisitor,todaystaff,repeatedVisitor,newVisitor =  wifi_logic_set(
            mac_address,\
227                     todayvisitor,todaystaff,historystaff,historyvisitor)
228     df_each = ace2.pd.DataFrame()
229     df_each['state'] = data_wifi.loc[data_wifi['premisename'] ==
            premisename]['state']
230     df_each['premisename'] = premisename
231     df_each['new_vis_cur'] = newVisitor
232     df_each['rep_vis_cur'] = repeatedVisitor
233
234
235     # UPdateing macs address to each premise
236     posttodaystaff(premisename,todaystaff,url,port,date)
237     posttodayvisitor(premisename,todayvisitor,url,port,date)
238
239     # In the case of No Data in History , just dump today data as his
240     if len(historystaff) == 0:
241         posthistorystaff(premisename,todaystaff,url,port)
242     if len(historyvisitor) == 0:
243         posthistoryvisitor(premisename,todayvisitor,url,port)
244     return df_each
245 def wifprocessing(tmpoints,mal_predata,state_predata,time_to_write,date):
```

```
246     tmpoint_detail = tmpoints.to_json(orient='records')
247     tmpoint_detail = ace2.json.loads(tmpoint_detail)
248
249     if len(mal_predata) > 0:
250         timestamp_check = int(mal_predata.iloc[0]['timestamp'])
251         diff_time = int(time_to_write)-timestamp_check
252         # print("Time Diff: ",diff_time)
253
254     # State Operations
255     state_group = tmpoints.groupby(by=tmpoints.state,as_index=False)
256     state_detail = list()
257     for key,value in state_group:
258         group = ace2.pd.DataFrame(value)
259         state = dict()
260         state["sta_name"] = group.iloc[0]['state']
261         state["sta_new_vis_cur"] = sum(group['new_vis_cur'])
262         state["sta_rep_vis_cur"] = sum(group['rep_vis_cur'])
263         if len(state_predata) > 0:
264             predata = state_predata.loc[(state_predata['sta_name'] ==
                    state["sta_name"])]
265
266             if len(predata) > 0 :
267                 pre_new_vis = int(predata.iloc[0]['sta_new_vis_tol'])
268                 pre_rep_vis = int(predata.iloc[0]['sta_rep_vis_tol'])
269                 state['sta_rep_vis_tol'] = state['sta_rep_vis_cur'] +
                        pre_rep_vis
270                 state['sta_new_vis_tol'] = state['sta_new_vis_cur'] +
                        pre_new_vis
271             else:
272                 state['sta_rep_vis_tol'] = state['sta_new_vis_cur']
273                 state['sta_new_vis_tol'] = state['sta_new_vis_cur']
274         else:
275             state['sta_rep_vis_tol'] = state['sta_rep_vis_cur']
276             state['sta_new_vis_tol'] = state['sta_new_vis_cur']
277         state_detail.append(state)
278
279     state_detail_json = json.dumps(state_detail)
280     state_detail_json = json.loads(state_detail_json)
281
282     data_obj = dict()
283     data_obj['state_detail'] = state_detail_json
284     data_obj['tmpoint_detail'] = tmpoint_detail
285     data_obj["mal_new_vis_cur"] = sum(tmpoints['new_vis_cur'])
286     data_obj["mal_rep_vis_cur"] = sum(tmpoints['rep_vis_cur'])
287     data_obj['timestamp'] = time_to_write
288     data_obj['date'] = date
289
290     if len(mal_predata) > 0 and diff_time == 300:
291         data_obj['mal_new_vis_tol'] = data_obj['mal_new_vis_cur'] + int(
                mal_predata.iloc[0]['mal_new_vis_tol'])
292         data_obj['mal_rep_vis_tol'] = data_obj['mal_rep_vis_cur'] + int(
                mal_predata.iloc[0]['mal_rep_vis_tol'])
293     elif len(mal_predata) > 0 and diff_time > 300:
```

```
294         data_obj['mal_new_vis_tol'] = data_obj['mal_new_vis_cur'] + int(
                mal_predata.iloc[0]['mal_new_vis_tol'])
295         data_obj['mal_rep_vis_tol'] = data_obj['mal_rep_vis_cur'] + int(
                mal_predata.iloc[0]['mal_rep_vis_tol'])
296         data_obj['timestamp'] = int(time_to_write) + diff_time
297         data_obj['date'] = date
298     elif len(mal_predata) > 0 and diff_time < 300:
299         data_obj['mal_new_vis_tol'] = data_obj['mal_new_vis_cur'] + int(
                mal_predata.iloc[0]['mal_new_vis_tol'])
300         data_obj['mal_rep_vis_tol'] = data_obj['mal_rep_vis_cur'] + int(
                mal_predata.iloc[0]['mal_rep_vis_tol'])
301         data_obj['timestamp'] = int(time_to_write) + 300
302         data_obj['date'] = date
303     else:
304         data_obj['mal_new_vis_tol'] = data_obj['mal_new_vis_cur']
305         data_obj['mal_rep_vis_tol'] = data_obj['mal_rep_vis_cur']
306 
307     return data_obj
308 
309 # Operations without multiprocessing
310 def packoperation_WNMP(companydetails,data,time_to_write,date,url,port):
311     for each in companydetails:
312         data_each = data[data['premisename'].isin(each['wifipremises'])]
313         if len(data_each) == 0:
314             continue
315         else:
316             column_flag =False
317             tmpoints = ace2.pd.DataFrame()
318             for item in each['wifipremises']:
319                 print(item)
320                 data_wifi = data.loc[data['premisename'] ==item]
321                 mal_predata,state_predata,tmpoint_predata =
                        getwifilastdata(date,each['name'],url,port)
322                 if len(data_wifi) > 0:
323                     df_each = wifioperation(data_wifi,item,url,port,date)
324 
325                     if len(tmpoint_predata) > 0 :
326                         predata_tmpoint = tmpoint_predata.loc[(
                                tmpoint_predata['premisename'] == item)]
327                         if len(predata_tmpoint) > 0:
328                             pre_new_vis = int(predata_tmpoint.iloc[0]['
                                    new_vis_tol'])
329                             pre_rep_vis = int(predata_tmpoint.iloc[0]['
                                    rep_vis_tol'])
330                             df_each['new_vis_tol'] =  df_each['new_vis_cur
                                    '] + pre_new_vis
331                             df_each['rep_vis_tol'] =  df_each['rep_vis_cur
                                    '] + pre_rep_vis
332 
333                         else:
334                             df_each['new_vis_tol'] =  df_each['new_vis_cur
                                    ']
335                             df_each['rep_vis_tol'] =  df_each['rep_vis_cur
                                    ']
```

```
336                     else:
337                         df_each['new_vis_tol'] =  df_each['new_vis_cur']
338                         df_each['rep_vis_tol'] =  df_each['rep_vis_cur']
339 
340                     if column_flag == False:
341                         tmpoints = df_each
342                         column_flag = True
343                     else:
344                         tmpoints = tmpoints.append(df_each,ignore_index=
                               True)
345                 # if wifi data distraction happen in specific tmpoint
346                 else:
347                     if len(tmpoint_predata) > 0 :
348                         predata_tmpoint = tmpoint_predata.loc[(
                               tmpoint_predata['premisename'] == item)]
349                         if len(predata_tmpoint) > 0:
350                             tmpoints = tmpoints.append(df_each,
                                   ignore_index=True)
351 
352             data_obj = wifprocessing(tmpoints,mal_predata,state_predata,
                   time_to_write,date)
353             if len(data_obj) > 0:
354                 postwifidaily(data_obj, each['name'],url,port)
355 
356 
357 # Operations without multiprocessing
358 # def worker(data,companyname,wifipremises,time_to_write,date,data_pack,
       url,port):
359 #     data_each = data[data['premisename'].isin(wifipremises)]
360 #     if len(data_each) > 0:
361 #         column_flag =False
362 #         tmpoints = ace2.pd.DataFrame()
363 #         for item in wifipremises:
364 #             print(item)
365 #             data_wifi = data.loc[data['premisename'] ==item]
366 #             if len(data_wifi) > 0:
367 #                 mal_predata,state_predata,tmpoint_predata =
       getwifilastdata(date,companyname)
368 #                 df_each = wifioperation(data_wifi,item,url,port)
369 #
370 #                 if len(tmpoint_predata) > 0 :
371 #                     predata_tmpoint = tmpoint_predata.loc[(
       tmpoint_predata['premisename'] == item)]
372 #                     if len(predata_tmpoint) > 0:
373 #                         pre_new_vis = int(predata_tmpoint.iloc[0]['
       new_vis_tol'])
374 #                         pre_rep_vis = int(predata_tmpoint.iloc[0]['
       rep_vis_tol'])
375 #                         df_each['new_vis_tol'] =  df_each['new_vis_cur']
        + pre_new_vis
376 #                         df_each['rep_vis_tol'] =  df_each['rep_vis_cur']
        + pre_rep_vis
377 #
378 #                     else:
```

```
379 #                         df_each['new_vis_tol'] =  df_each['new_vis_cur']
380 #                         df_each['rep_vis_tol'] =  df_each['rep_vis_cur']
381 #                 else:
382 #                     df_each['new_vis_tol'] =  df_each['new_vis_cur']
383 #                     df_each['rep_vis_tol'] =  df_each['rep_vis_cur']
384 #
385 #                 if column_flag == False:
386 #                     tmpoints = df_each
387 #                     column_flag = True
388 #                 else:
389 #                     tmpoints = tmpoints.append(df_each,ignore_index=True
    )
390 #             # if wifi data distraction happen in specific tmpoint
391 #             else:
392 #                 if len(tmpoint_predata) > 0 :
393 #                     predata_tmpoint = tmpoint_predata.loc[(
    tmpoint_predata['premisename'] == item)]
394 #                     if len(predata_tmpoint) > 0:
395 #                         tmpoints = tmpoints.append(df_each,ignore_index=
    True)
396 #
397 #
398 #         data_obj = wifprocessing(tmpoints,mal_predata,state_predata,
    time_to_write,date)
399 #         data_pack[companyname] = data_obj
400 # def packoperation_WMP(companydetails,data,time_to_write,date):
401 #     manager = mp.Manager()
402 #     data_pack = manager.dict()
403 #     totalProcess = []
404 #     # Start proceses
405 #     for each in companydetails:
406 #         data_each = data[data['premisename'].isin(each['wifipremises'])]
407 #         p = mp.Process(target=worker, args=(data_each,each['name'],each
    ['wifipremises'],time_to_write,date,data_pack))
408 #         totalProcess.append(p)
409 #         p.start()
410 #     for proc in totalProcess:
411 #         proc.join()
412 #     # Get the return elements from each process
413 #     final_data_pack = []
414 #     for each in companydetails:
415 #         if each['name'] in data_pack.keys():
416 #             data_each = dict()
417 #             data_each['companyname'] = each['name']
418 #             data_each['data'] = data_pack[each['name']]
419 #             final_data_pack.append(data_each)
420 #     return final_data_pack
421
422
423 def main (date,hour,min_,loop=False):
424
425
426     source = ace2.read_json('../input.json')
427     url = source['url']
```

```
428     port = source['port']
429     companydetails = ace2.pickpremisebycom(url,port)
430     premisedata = ace2.getpremises(url,port)
431     time_to_write_ = date + " " + hour + ":" + min_
432     print(time_to_write_)
433     time_to_write = ace2.datetime.strptime(time_to_write_,"%Y%m%d %H:%M")
434     time_to_write = time_to_write.strftime('%s')
435 
436     if loop == True:
437         premises = []
438         for each in companydetails:
439             for item in each['wifipremises']:
440                 premises.append(item)
441         data = getwifidata(date,hour,min_,premises,url,port)
442     else:
443         data = getwifi5mindata(date,url,port)
444     if len(data) > 0:
445         data = ace2.pd.merge(data, premisedata, on='premisename', how='
              left')
446         packoperation_WNMP(companydetails,data,time_to_write,date,url,port
              )
447 
448     else:
449         print('No Wifi Data')
```

ace2/wifi/___init___.py

```
1  __all__ = ['daily','weekly','monthly','appendeod','captivedaily','
       captiveweekly','captivemonthly']
2 
3  import ace2.wifi.daily
4  import ace2.wifi.weekly
5  import ace2.wifi.monthly
6  import ace2.wifi.appendeod
7  import ace2.wifi.captivedaily
8  import ace2.wifi.captiveweekly
9  import ace2.wifi.captivemonthly
```

ace2/wifi/monthly.py

```
1  import ace2
2  import os.path
3  import pandas as pd
4  import json
5  import requests as rq
6 
7  def getwifimonthlydata(companyname,month,url,port):
8      request = rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/monthlydata
           ?company='+companyname+'&month='+month, \
9               headers= {'Content-type': 'application/json'})
```

```
10      value = request.text
11      data = json.loads(value)
12      if data['error'] == False:
13          data = data['WifiMonthlylyData']
14          if len(data) == 0:
15
16              tmp_data = pd.DataFrame()
17              sta_data = pd.DataFrame()
18              mal_data = pd.DataFrame()
19              return (mal_data,sta_data,tmp_data)
20          else:
21              columns_flag = False
22              for each in data:
23                  dailydata = each['data']
24                  dailytmpoints = dailydata['tmpoint_detail']
25                  dailystates = dailydata['state_detail']
26                  dailystates = pd.io.json.json_normalize(dailystates)
27                  dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
28                  dailymal = pd.io.json.json_normalize(dailydata)
29                  dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                        inplace = True)
30                  if columns_flag == False:
31                      tmp_data = dailytmpoints
32                      sta_data = dailystates
33                      mal_data = dailymal
34                      columns_flag= True
35                  else:
36
37                      tmp_data = tmp_data.append(dailytmpoints, ignore_index
                            =True)
38                      sta_data = sta_data.append(dailystates, ignore_index=
                            True)
39                      mal_data = mal_data.append(dailymal, ignore_index=True
                            )
40              return (mal_data,sta_data,tmp_data)
41
42  def wifiOps(companyname,month,url,port):
43      mal_data,sta_data,tmp_data=getwifimonthlydata(companyname,month,url,
            port)
44      if len(mal_data) > 0 and len(sta_data) > 0 and len(tmp_data) > 0:
45          # General Processing
46          # Malaysia Operations
47
48          weekno = ace2.datetime.today().isocalendar()[1]
49          year = ace2.datetime.today().strftime("%Y")
50          unique_key = year+month
51          print(unique_key)
52
53
54          mal_data_cal = dict()
55          mal_data_cal['date'] = unique_key
56          mal_data_cal['mal_new_vis_tol'] = sum(mal_data['mal_new_vis_tol'])
57          mal_data_cal['mal_rep_vis_tol'] = sum(mal_data['mal_rep_vis_tol'])
58
```

```
59          # States Operations
60          sta_data_group = sta_data.groupby(by=sta_data.sta_name,as_index=
                False)
61          sta_data_detail = list()
62          for key,value in sta_data_group:
63              group = pd.DataFrame(value)
64              sta_data_cal = dict()
65              sta_data_cal['sta_name'] = group.iloc[0]['sta_name']
66              sta_data_cal['sta_new_vis_tol'] = sum(group['sta_new_vis_tol'
                    ])
67              sta_data_cal['sta_rep_vis_tol'] = sum(group['sta_rep_vis_tol'
                    ])
68              sta_data_detail.append(sta_data_cal)
69          mal_data_cal['state_detail'] = sta_data_detail
70
71          # TMpoints Operations
72          tmpoints_group = tmp_data.groupby(by=tmp_data.premisename,as_index
                =False)
73          tmp_detail = list()
74
75          for key,value in tmpoints_group:
76              group = pd.DataFrame(value)
77              tmpoint_cal = dict()
78              tmpoint_cal['premisename'] = group.iloc[0]['premisename']
79              tmpoint_cal['state'] = group.iloc[0]['state']
80
81              tmpoint_cal['new_vis_tol'] = sum(group['new_vis_tol'])
82              tmpoint_cal['rep_vis_tol'] = sum(group['rep_vis_tol'])
83              tmp_detail.append(tmpoint_cal)
84          mal_data_cal['tmpoint_detail'] = tmp_detail
85
86          postwifidaily(mal_data_cal, companyname,url,port)
87
88  def postwifidaily(data, companyname,url,port):
89      url = 'http://'+url+':'+port+'/ace/api/v1/wifi/monthly?company='+
            companyname
90      headers = {'Content-type': 'application/json'}
91      r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
92      print('postwifiweekly',r.text)
93
94  def main(month):
95      source = ace2.read_json('../input.json')
96      url = source['url']
97      port = source['port']
98      companydetails = ace2.pickpremisebycom(url,port)
99      for each in companydetails:
100         wifiOps(each['name'],month,url,port)
```

ace2/wifi/weekly.py

```
1   import ace2
2   import os.path
```

```
3  import pandas as pd
4  import json
5  import requests as rq
6
7  def getwifiweeklydata(companyname,url,port,weekno):
8      request = rq.get ('http://'+url+':'+port+'/ace/api/v1/wifi/weeklydata?
           company='+companyname+'&week='+weekno, \
9              headers= {'Content-type': 'application/json'})
10     value = request.text
11     data = json.loads(value)
12     if data['error'] == False:
13         data = data['WifiWeeklyData']
14         if len(data) == 0:
15
16             tmp_data = pd.DataFrame()
17             sta_data = pd.DataFrame()
18             mal_data = pd.DataFrame()
19             return (mal_data,sta_data,tmp_data)
20         else:
21             columns_flag = False
22             for each in data:
23                 dailydata = each['data']
24                 dailytmpoints = dailydata['tmpoint_detail']
25                 dailystates = dailydata['state_detail']
26                 dailystates = pd.io.json.json_normalize(dailystates)
27                 dailytmpoints = pd.io.json.json_normalize(dailytmpoints)
28                 dailymal = pd.io.json.json_normalize(dailydata)
29                 dailymal.drop(['tmpoint_detail','state_detail'], axis = 1,
                       inplace = True)
30                 if columns_flag == False:
31                     tmp_data = dailytmpoints
32                     sta_data = dailystates
33                     mal_data = dailymal
34                     columns_flag= True
35                 else:
36
37                     tmp_data = tmp_data.append(dailytmpoints, ignore_index
                           =True)
38                     sta_data = sta_data.append(dailystates, ignore_index=
                           True)
39                     mal_data = mal_data.append(dailymal, ignore_index=True
                           )
40             return (mal_data,sta_data,tmp_data)
41
42 def wifiOps(companyname,url,port,weekno):
43     mal_data,sta_data,tmp_data=getwifiweeklydata(companyname,url,port,
           weekno)
44     if len(mal_data) > 0 and len(sta_data) > 0 and len(tmp_data) > 0:
45         # General Processing
46         # Malaysia Operations
47
48
49         year = ace2.datetime.today().strftime("%Y")
50         unique_key = year+'-'+str(weekno)
```

```
51          print(unique_key)
52
53          mal_data_cal = dict()
54          mal_data_cal['date'] = unique_key
55          mal_data_cal['mal_new_vis_tol'] = sum(mal_data['mal_new_vis_tol'])
56          mal_data_cal['mal_rep_vis_tol'] = sum(mal_data['mal_rep_vis_tol'])
57
58          # States Operations
59          sta_data_group = sta_data.groupby(by=sta_data.sta_name,as_index=
                False)
60          sta_data_detail = list()
61          for key,value in sta_data_group:
62              group = pd.DataFrame(value)
63              sta_data_cal = dict()
64              sta_data_cal['sta_name'] = group.iloc[0]['sta_name']
65              sta_data_cal['sta_new_vis_tol'] = sum(group['sta_new_vis_tol'
                    ])
66              sta_data_cal['sta_rep_vis_tol'] = sum(group['sta_rep_vis_tol'
                    ])
67              sta_data_detail.append(sta_data_cal)
68          mal_data_cal['state_detail'] = sta_data_detail
69
70          # TMpoints Operations
71          tmpoints_group = tmp_data.groupby(by=tmp_data.premisename,as_index
                =False)
72          tmp_detail = list()
73
74          for key,value in tmpoints_group:
75              group = pd.DataFrame(value)
76              tmpoint_cal = dict()
77              tmpoint_cal['premisename'] = group.iloc[0]['premisename']
78              tmpoint_cal['state'] = group.iloc[0]['state']
79
80              tmpoint_cal['new_vis_tol'] = sum(group['new_vis_tol'])
81              tmpoint_cal['rep_vis_tol'] = sum(group['rep_vis_tol'])
82              tmp_detail.append(tmpoint_cal)
83          mal_data_cal['tmpoint_detail'] = tmp_detail
84
85          postwifidaily(mal_data_cal, companyname,url,port)
86
87  def postwifidaily(data, companyname,url,port):
88      url = 'http://'+url+':'+port+'/ace/api/v1/wifi/weekly?company='+
            companyname
89      headers = {'Content-type': 'application/json'}
90      r = ace2.rq.post(url, data=json.dumps(data), headers=headers)
91      print('postwifiweekly',r.text)
92
93  def main(weekno):
94      source = ace2.read_json('../input.json')
95      url = source['url']
96      port = source['port']
97      companydetails = ace2.pickpremisebycom(url,port)
98      for each in companydetails:
99          wifiOps(each['name'],url,port,weekno)
```