AADT_Estimator

October 31, 2017

1 AADT Estimator

1.0.1 Flow detector available count data date range function

- find count data date range of a flow detector

```
In [1]: from utility import db_connect, query2csv
        from settings import  DBNAME, DBPASS, DBUSER, DBHOST

        def get_date_range(flow_detector_id):
            query = """
        select
          to_char(min(start_time), 'YYYY-MM-DD HH24:MI:SS') as beginning,
          to_char(max(start_time), 'YYYY-MM-DD HH24:MI:SS') as end
        from
          baa_ex_sus.data
        where
          flow_detector_id = {0}
        """.format(flow_detector_id)
            conn = db_connect()
            with conn:
                with conn.cursor() as curs:
                    curs.execute(query)
                    rows = curs.fetchall()
                    return (rows[0][0],rows[0][1])
```

```
In [2]: flow_detector_id = 1295
        get_date_range(flow_detector_id)
```

```
Out[2]: ('2015-07-27 16:00:00', '2017-10-02 00:00:00')
```

- Get 4 weeks (2016-07-31 - 2016-08-27, within the date range) of count data in 15min bin for flow detector 1295
- 4 weeks of raw data in **UTC timezone**

```
In [3]: from utility import db_connect, query2csv
        from settings import  DBNAME, DBPASS, DBUSER, DBHOST
```

```
        qsql="""
        select
          *
        from
          bike_ped.data
        where
          flow_detector_id = 1295
          and start_time>='2016-07-31'
          and end_time <'2016-08-28'
        """
        csvfile='count_data_1295_4weeks.csv'
        query2csv(qsql,csvfile)
```

```
<IPython.core.display.HTML object>
```

```
In [4]: from utility import db_connect, query2csv
        from settings import  DBNAME, DBPASS, DBUSER, DBHOST

        qsql="""
        select
          *
        from
          baa_ex_sus.data
        where
          flow_detector_id = 1295
          and start_time>='2016-07-31'
          and end_time <'2016-08-28'
        """
        csvfile='count_data_1295_localtime_4weeks.csv'
        query2csv(qsql,csvfile)
```

```
<IPython.core.display.HTML object>
```

- get the 4 weeks of raw data and convert the timezone to local time zone and aggregate volume to 1 hour interval

```
In [5]: from utility import db_connect, query2csv
        from settings import  DBNAME, DBPASS, DBUSER, DBHOST

        qsql="""
        with fltz as (
          select
            flow_detector_id,
            bike_ped.get_flow_detector_timezone(flow_detector_id)
              as timezone
          from
            bike_ped.flow_detectors
```

```
  where
    flow_detector_id = 1295
  )
  select
    bpd.flow_detector_id,
    bpd.upload_id,
    date_trunc('hour', bpd.start_time) as start_time_utc,
    date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      as start_time,
    fltz.timezone as timezone,
    date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      +'1 hour'::interval as end_time,
    '1 hour'::INTERVAL as measure_period,
    sum(volume) as volume
from
    bike_ped.data as bpd inner join fltz using(flow_detector_id)
where
    measure_period='00:15:00'
    and bpd.flow_detector_id = 1295
    and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      >='2016-07-31'
    and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      <'2016-08-28'
    group by flow_detector_id, upload_id, date_trunc('hour', start_time),
      fltz.timezone,
      date_trunc('hour', bpd.start_time at time zone fltz.timezone)
    having count(start_time) = 4
"""
csvfile='count_data_1295_4weeks_localtz_1hr.csv'
query2csv(qsql,csvfile)
```

```
<IPython.core.display.HTML object>
```

- get the 4 weeks of raw data and aggregate to 1 day interval

```
In [8]: from utility import db_connect, query2csv
        from settings import  DBNAME, DBPASS, DBUSER, DBHOST

        qsql="""
        with fltz as (
          select
            flow_detector_id,
            bike_ped.get_flow_detector_timezone(flow_detector_id)
              as timezone
          from
            bike_ped.flow_detectors
          where
```

```
                flow_detector_id = 1295
    ),
  hrly as (
    select
      bpd.flow_detector_id,
      bpd.upload_id,
      date_trunc('hour', bpd.start_time) as start_time_utc,
      date_trunc('hour', bpd.start_time at time zone fltz.timezone)
        as start_time,
      fltz.timezone as timezone,
      date_trunc('hour', bpd.start_time at time zone fltz.timezone)
        +'1 hour'::interval as end_time,
      '1 hour'::INTERVAL as measure_period,
      sum(volume) as volume
  from
      bike_ped.data as bpd inner join fltz using(flow_detector_id)
  where
      measure_period='00:15:00'
      and bpd.flow_detector_id = 1295
      and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
        >='2016-07-31'
      and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
        <'2016-08-28'
      group by flow_detector_id, upload_id,
        date_trunc('hour', start_time), fltz.timezone,
        date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      having count(start_time) = 4
  )
  select
      flow_detector_id,
      date_trunc('day', start_time) as date,
      sum(volume) as count,
      extract(dow from start_time) as dow
  from
      hrly
  group by 1,2,4
  order by 2
  """
  csvfile='count_data_1295_4weeks_localtz_1day.csv'
  query2csv(qsql,csvfile)
```

```
<IPython.core.display.HTML object>
```

1.0.2 Choose Factors

1.0.3 Calculate WWI from the data

In [9]:
```
from utility import db_connect, query2csv
from settings import  DBNAME, DBPASS, DBUSER, DBHOST

def calculate_wwi(flow_detector_id, start_time, end_time):
    query = """
with fltz as (
        select
          flow_detector_id,
          bike_ped.get_flow_detector_timezone(flow_detector_id) as timezone
        from
          bike_ped.flow_detectors
        where
          flow_detector_id = {0}
  ),
hrly as (
  select
    bpd.flow_detector_id,
    bpd.upload_id,
    date_trunc('hour', bpd.start_time) as start_time_utc,
    date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      as start_time,
    fltz.timezone as timezone,
    date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      +'1 hour'::interval as end_time,
    '1 hour'::INTERVAL as measure_period,
    sum(volume) as volume
from
    bike_ped.data as bpd inner join fltz using(flow_detector_id)
where
    measure_period='00:15:00'
    and bpd.flow_detector_id = {0}
    and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      >='{1}'
    and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      <'{2}'
    group by flow_detector_id, upload_id,
      date_trunc('hour', start_time),
      fltz.timezone ,
      date_trunc('hour', bpd.start_time at time zone fltz.timezone)
    having count(start_time) = 4
),
daily as (
select
    flow_detector_id,
```

```
        date_trunc('day', start_time) as date,
        sum(volume) as count,
        extract(dow from start_time) as dow
    from
        hrly
        group by 1,2,4
        --Restriction: Do not estimate AADT for days with <22hrs or >26 hours.
        having count(*) > 22 and count(*) < 26
    ),
    -- weekend volume
    V_we as (
      select
        avg(count) as vwe
      from
        daily
      where
        dow in (0, 6)
    ),
    -- weekday volume
    V_wd as (
      select
        avg(count) as vwd
      from
        daily
      where
        dow in (1,2,3,4,5)
    )
    select round(V_we.vwe/V_wd.vwd, 2) as wwi from V_we, V_wd
    """.format(flow_detector_id, start_time, end_time)
        conn = db_connect()
        with conn:
            with conn.cursor() as curs:
                curs.execute(query)
                rows = curs.fetchall()
                return (rows[0][0])
```

```
In [10]: start_time = '2016-07-31'
         end_time = '2016-08-28'
         flow_detector_id = 1295
         wwi = calculate_wwi(flow_detector_id, start_time, end_time)

         if wwi<=0.8:
             wwitype='Weekday Commute'
         elif wwi>0.8 or wwi <=1.2:
             wwitype='Weekly Multipurpose'
         elif wwi > 1.2:
             wwitype='Weekend Multipurpose'
         print('The traffic pattern is {0} with wwi={1}'.format(wwitype, wwi))
```

```
The traffic pattern is Weekly Multipurpose with wwi=0.87
```

1.0.4 Compute the estimated AADT

- We are using data for flow detector **1295**
- location city is **San Diego**
- 4 weeks of data start from **2016-07-31** to **2016-08-27**
- mode is **bicycle**
- The follow query will return **estimated AADT for each day** for this flow detector in specified date range

```
In [11]: from utility import db_connect, query2csv
         from settings import  DBNAME, DBPASS, DBUSER, DBHOST

         qsql = """
         with d as (
           select generate_series(0,6) as dayofweek
         ),
         m as (
           select generate_series(1,12) as month
         ),
         -- v_ijmy:Compute an average by day of week for each month.
         v_ijmy as (
           select
               baadv.analysis_area_id,
               date_part('year', baadv.date) as year,
               avg(baadv.volume)::bigint as volume_i,
               avg(baadv.volume) as volume,
               d.dayofweek,
               m.month
           from
               baa_ex_sus.analysis_areas_daily_volume as baadv,
               d,
               m
           where
               extract(dow from baadv.date) in (d.dayofweek)
               AND date_part('month', baadv.date) = m.month
               group by baadv.analysis_area_id, year, d.dayofweek, m.month
         ),
         -- madt: average volume each month, each year for sites
         madt as (
           select
               analysis_area_id,
               month,
               year,
               avg(volume)::bigint as volume_i,
               avg(volume) as volume
```

```
  from
      v_ijmy
      group by analysis_area_id, year, month
      having count(dayofweek)=7 -- having 7 days of data each week
),
AADT as (
select
  analysis_area_id,
  year,
  avg(volume)::bigint as AADT_i,
  round(avg(volume), 2) as AADT
from madt
  group by analysis_area_id, year
  having count(month) = 12 -- having 12 months of data
),
-- daily_exclude_holiday: daily counts for sites excluding holidays
daily_exclude_holiday as (
select
 baaad.analysis_area_id,
 baaad.date,
 baaad.volume,
 date_part('month', baaad.date) as month,
 date_part('dow', baaad.date) as dow
from
  baa_ex_sus.analysis_areas_daily_volume as baaad
  left join baa.holidays as baahd on baaad.date::date = baahd.holiday_date
where
  baahd.holiday_id is null
  group by 1,2,3
),
V_jmyl_exclude_holiday as (
  select
      baadv.analysis_area_id,
      date_part('year', baadv.date) as year,
      avg(baadv.volume) as volume,
      d.dayofweek,
      m.month
  from
      daily_exclude_holiday as baadv,
      d,
      m
  where
      extract(dow from baadv.date) in (d.dayofweek)
      AND date_part('month', baadv.date) = m.month
      group by baadv.analysis_area_id, year, d.dayofweek, m.month
),
-- 84 factors volume count should exclude holiday weeks
factor84 as (
```

```
select
  v_jmyl_nh.analysis_area_id,
  v_jmyl_nh.volume as v_jmyl,
  AADT.aadt as aadt,
  round(v_jmyl_nh.volume/aadt::numeric, 2) as f_jmys,
  v_jmyl_nh.dayofweek,
  v_jmyl_nh.month,
  v_jmyl_nh.year
from
  V_jmyl_exclude_holiday as v_jmyl_nh inner join AADT
    using(analysis_area_id, year)
where
  AADT.AADT <> 0
),
V_we as (
select
  baadv.analysis_area_id,
  avg(baadv.volume) vwe
from
  baa_ex_sus.analysis_areas_daily_volume as baadv
where
  extract(dow from baadv.date) in (0,6)
  group by baadv.analysis_area_id
),
V_wd as (
select
  baadv.analysis_area_id,
  avg(baadv.volume) vwd
from
  baa_ex_sus.analysis_areas_daily_volume as baadv
where
  extract(dow from baadv.date) in (1,2,3,4,5)
  group by baadv.analysis_area_id
),
grouping as (
select
  V_we.analysis_area_id,
  round(V_we.vwe, 2) as V_we,
  round(V_wd.vwd, 2) as V_wd,
  round(V_we.vwe/V_wd.vwd, 2) as wwi,
  case
     when (round(V_we.vwe/V_wd.vwd, 2) <= 0.8)
       then 'Weekday Commute'
     when (round(V_we.vwe/V_wd.vwd, 2) >  1.2)
       then 'Weekend Multipurpose'
     ELSE 'Weekly Multipurpose'
  END as grouping
from
```

```
  V_we inner join V_wd using (analysis_area_id)
),
wwi as (
select
  grouping.analysis_area_id,
  baaa.mode,
  baaa.analysis_area_name,
  baaa.analysis_area_regions_id,
  grouping.v_we,
  grouping.v_wd,
  grouping.wwi,
  grouping.grouping as weekly_group
from
   grouping inner join baa.analysis_areas as baaa
     using(analysis_area_id)
),
factorgrp as (
select
  ar.analysis_area_name as city,
  wwi.mode,
  wwi.weekly_group,
  array_agg(wwi.analysis_area_id order by analysis_area_id)
    as analysis_area_id_list
from
  wwi, baa.analysis_area_regions as ar
where
  ar.analysis_area_regions_id = wwi.analysis_area_regions_id
  group by 1,2,3
), f84_wwi as (
  select
    fg.city,
    fg.weekly_group,
    fg.mode,
    fg.analysis_area_id_list,
    f84.dayofweek,
    f84.month,
    f84.year,
    round(avg(f84.f_jmys), 2) as f_jmys_avg
  from
    factor84 as f84 inner join factorgrp as fg
    on f84.analysis_area_id = Any(fg.analysis_area_id_list::int[])
    group by
    fg.city,
    fg.weekly_group,
    fg.mode,
    fg.analysis_area_id_list,
    f84.dayofweek,
    f84.month,
```

```
    f84.year
    order by fg.city,
    fg.weekly_group,
    fg.mode,f84.year, f84.month, f84.dayofweek
    ),
-- start aadt estimator for a flow detector
fltz as (
        select
          flow_detector_id,
          bike_ped.get_flow_detector_timezone(flow_detector_id)
            as timezone
        from
          bike_ped.flow_detectors
        where
          flow_detector_id = 1295
  ),
hrly as (
  select
    bpd.flow_detector_id,
    bpd.upload_id,
    date_trunc('hour', bpd.start_time) as start_time_utc,
    date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      as start_time,
    fltz.timezone as timezone,
    date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      +'1 hour'::interval as end_time,
    '1 hour'::INTERVAL as measure_period,
    sum(volume) as volume
from
    bike_ped.data as bpd inner join fltz using(flow_detector_id)
where
    measure_period='00:15:00'
    and bpd.flow_detector_id = 1295
    and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      >='2016-07-31'
    and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      <'2016-08-28'
    group by flow_detector_id, upload_id,
      date_trunc('hour', start_time), fltz.timezone ,
      date_trunc('hour', bpd.start_time at time zone fltz.timezone)
    having count(start_time) = 4
),
daily as (
select
    flow_detector_id,
    date_trunc('day', start_time) as date,
    sum(volume) as count,
    extract(dow from start_time) as dow
```

```
from
    hrly
    group by 1,2,4
    --Restriction:Do not estimate AADT for days with <22hrs or >26 hours.
    having count(*) > 22 and count(*) < 26
),
-- weekend volume
vwe as (
  select
    avg(count) as vwe
  from
    daily
  where
    dow in (0, 6)
),
-- weekday volume
vwd as (
  select
    avg(count) as vwd
  from
    daily
  where
    dow in (1,2,3,4,5)
),
vwwi as (
select
  daily.flow_detector_id,
  daily.date,
  date_part('month', daily.date) as month,
  date_part('year', daily.date) as year,
  daily.count,
  daily.dow,
  --round(vwe.vwe,2) as vwe,
  --round(vwd.vwd,2) as vwd,
  round(vwe.vwe/vwd.vwd, 2) as wwi,
  case
    when (round(vwe.vwe/vwd.vwd, 2) <= 0.8) then 'Weekday Commute'
    when (round(vwe.vwe/vwd.vwd, 2) >  1.2) then 'Weekend Multipurpose'
    ELSE 'Weekly Multipurpose'
  END as grouping
  from vwe, vwd, daily
)
select
  vwwi.flow_detector_id,
  vwwi.date,
  vwwi.month,
  vwwi.year,
  vwwi.count,
```

```
            vwwi.grouping,
            f84.mode,
            round(vwwi.count/f84.f_jmys_avg, 2) as est_aadt
          from vwwi,f84_wwi as f84
          where
            f84.city = 'San Diego'
            and f84.month = vwwi.month
            and f84.year = vwwi.year
            and f84.weekly_group = vwwi.grouping
            and f84.dayofweek = vwwi.dow
            and f84.mode = 'bicycle'
            order by vwwi.date
          """
          csvfile='est_aadt_flow_detector_1295_4weeks.csv'
          query2csv(qsql,csvfile)
```

```
<IPython.core.display.HTML object>
```

- Estimated AADT for sample data

```
In [12]: from utility import db_connect, query2csv
         from settings import  DBNAME, DBPASS, DBUSER, DBHOST

         def calculate_est_aadt(flow_detector_id, start_time,
                                end_time, city, mode):
             query = """
         with d as (
           select generate_series(0,6) as dayofweek
         ),
         m as (
           select generate_series(1,12) as month
         ),
         -- v_ijmy:Compute an average by day of week for each month.
         v_ijmy as (
           select
               baadv.analysis_area_id,
               date_part('year', baadv.date) as year,
               avg(baadv.volume)::bigint as volume_i,
               avg(baadv.volume) as volume,
               d.dayofweek,
               m.month
           from
               baa_ex_sus.analysis_areas_daily_volume as baadv,
               d,
               m
           where
               extract(dow from baadv.date) in (d.dayofweek)
```

```
      AND date_part('month', baadv.date) = m.month
      group by baadv.analysis_area_id, year, d.dayofweek, m.month
),
-- madt: average volume each month, each year for sites
madt as (
  select
      analysis_area_id,
      month,
      year,
      avg(volume)::bigint as volume_i,
      avg(volume) as volume
  from
      v_ijmy
      group by analysis_area_id, year, month
      having count(dayofweek)=7 -- having 7 days of data each week
),
AADT as (
select
  analysis_area_id,
  year,
  avg(volume)::bigint as AADT_i,
  round(avg(volume), 2) as AADT
from madt
  group by analysis_area_id, year
  having count(month) = 12 -- having 12 months of data
),
-- daily_exclude_holiday: daily counts for sites excluding holidays
daily_exclude_holiday as (
select
 baaad.analysis_area_id,
 baaad.date,
 baaad.volume,
 date_part('month', baaad.date) as month,
 date_part('dow', baaad.date) as dow
from
  baa_ex_sus.analysis_areas_daily_volume as baaad
  left join baa.holidays as baahd
    on baaad.date::date = baahd.holiday_date
where
  baahd.holiday_id is null
  group by 1,2,3
),
V_jmyl_exclude_holiday as (
  select
      baadv.analysis_area_id,
      date_part('year', baadv.date) as year,
      avg(baadv.volume) as volume,
      d.dayofweek,
```

```
      m.month
  from
      daily_exclude_holiday as baadv,
      d,
      m
  where
      extract(dow from baadv.date) in (d.dayofweek)
      AND date_part('month', baadv.date) = m.month
      group by baadv.analysis_area_id, year, d.dayofweek, m.month
),
-- 84 factors volume count should exclude holiday weeks
factor84 as (
select
  v_jmyl_nh.analysis_area_id,
  v_jmyl_nh.volume as v_jmyl,
  AADT.aadt as aadt,
  round(v_jmyl_nh.volume/aadt::numeric, 2) as f_jmys,
  v_jmyl_nh.dayofweek,
  v_jmyl_nh.month,
  v_jmyl_nh.year
from
  V_jmyl_exclude_holiday as v_jmyl_nh inner join AADT
    using(analysis_area_id, year)
where
  AADT.AADT <> 0
),
V_we as (
select
  baadv.analysis_area_id,
  avg(baadv.volume) vwe
from
  baa_ex_sus.analysis_areas_daily_volume as baadv
where
  extract(dow from baadv.date) in (0,6)
  group by baadv.analysis_area_id
),
V_wd as (
select
  baadv.analysis_area_id,
  avg(baadv.volume) vwd
from
  baa_ex_sus.analysis_areas_daily_volume as baadv
where
  extract(dow from baadv.date) in (1,2,3,4,5)
  group by baadv.analysis_area_id
),
grouping as (
select
```

```
  V_we.analysis_area_id,
  round(V_we.vwe, 2) as V_we,
  round(V_wd.vwd, 2) as V_wd,
  round(V_we.vwe/V_wd.vwd, 2) as wwi,
  case
     when (round(V_we.vwe/V_wd.vwd, 2) <= 0.8)
       then 'Weekday Commute'
     when (round(V_we.vwe/V_wd.vwd, 2) >  1.2)
       then 'Weekend Multipurpose'
     ELSE 'Weekly Multipurpose'
  END as grouping
from
  V_we inner join V_wd using (analysis_area_id)
),
wwi as (
select
  grouping.analysis_area_id,
  baaa.mode,
  baaa.analysis_area_name,
  baaa.analysis_area_regions_id,
  grouping.v_we,
  grouping.v_wd,
  grouping.wwi,
  grouping.grouping as weekly_group
from
   grouping inner join baa.analysis_areas
     as baaa using(analysis_area_id)
),
factorgrp as (
select
  ar.analysis_area_name as city,
  wwi.mode,
  wwi.weekly_group,
  array_agg(wwi.analysis_area_id order by analysis_area_id)
    as analysis_area_id_list
from
  wwi, baa.analysis_area_regions as ar
where
  ar.analysis_area_regions_id = wwi.analysis_area_regions_id
  group by 1,2,3
), f84_wwi as (
  select
    fg.city,
    fg.weekly_group,
    fg.mode,
    fg.analysis_area_id_list,
    f84.dayofweek,
    f84.month,
```

```
    f84.year,
    round(avg(f84.f_jmys), 2) as f_jmys_avg
  from
    factor84 as f84 inner join factorgrp as fg
    on f84.analysis_area_id = Any(fg.analysis_area_id_list::int[])
    group by
    fg.city,
    fg.weekly_group,
    fg.mode,
    fg.analysis_area_id_list,
    f84.dayofweek,
    f84.month,
    f84.year
    order by fg.city,
    fg.weekly_group,
    fg.mode,f84.year, f84.month, f84.dayofweek
    ),
-- start aadt estimator for a flow detector
fltz as (
   select
     flow_detector_id,
     bike_ped.get_flow_detector_timezone(flow_detector_id)
       as timezone
     from
       bike_ped.flow_detectors
     where
       flow_detector_id = {0}
  ),
hrly as (
  select
    bpd.flow_detector_id,
    bpd.upload_id,
    date_trunc('hour', bpd.start_time) as start_time_utc,
    date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      as start_time,
    fltz.timezone as timezone,
    date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      +'1 hour'::interval as end_time,
    '1 hour'::INTERVAL as measure_period,
    sum(volume) as volume
from
    bike_ped.data as bpd inner join fltz using(flow_detector_id)
where
    measure_period='00:15:00'
    and bpd.flow_detector_id = {0}
    and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
      >='{1}'
    and date_trunc('hour', bpd.start_time at time zone fltz.timezone)
```

```
      <'{2}'
    group by flow_detector_id, upload_id,
      date_trunc('hour', start_time), fltz.timezone ,
      date_trunc('hour', bpd.start_time at time zone fltz.timezone)
    having count(start_time) = 4
),
daily as (
select
    flow_detector_id,
    date_trunc('day', start_time) as date,
    sum(volume) as count,
    extract(dow from start_time) as dow
from
    hrly
    group by 1,2,4
    --Restriction:Do not estimate AADT for days with <22hrs or >26 hours.
    having count(*) > 22 and count(*) < 26
),
-- weekend volume
vwe as (
  select
    avg(count) as vwe
  from
    daily
  where
    dow in (0, 6)
),
-- weekday volume
vwd as (
  select
    avg(count) as vwd
  from
    daily
  where
    dow in (1,2,3,4,5)
),
vwwi as (
select
  daily.flow_detector_id,
  daily.date,
  date_part('month', daily.date) as month,
  date_part('year', daily.date) as year,
  daily.count,
  daily.dow,
  --round(vwe.vwe,2) as vwe,
  --round(vwd.vwd,2) as vwd,
  round(vwe.vwe/vwd.vwd, 2) as wwi,
  case
```

```
            when (round(vwe.vwe/vwd.vwd, 2) <= 0.8)
              then 'Weekday Commute'
            when (round(vwe.vwe/vwd.vwd, 2) >  1.2)
              then 'Weekend Multipurpose'
            ELSE 'Weekly Multipurpose'
          END as grouping
          from vwe, vwd, daily
        ),
        est_aadt as (
        select
          vwwi.flow_detector_id,
          vwwi.date,
          vwwi.month,
          vwwi.year,
          vwwi.count,
          vwwi.grouping,
          f84.mode,
          round(vwwi.count/f84.f_jmys_avg, 2) as est_aadt
        from vwwi,f84_wwi as f84
        where
          f84.city = '{3}'
          and f84.month = vwwi.month
          and f84.year = vwwi.year
          and f84.weekly_group = vwwi.grouping
          and f84.dayofweek = vwwi.dow
          and f84.mode = '{4}'
          order by vwwi.date
          )
          select
          round(avg(est_aadt), 2) as estaadt
          from
            est_aadt
        """.format(flow_detector_id, start_time, end_time, city, mode)
            conn = db_connect()
            with conn:
                with conn.cursor() as curs:
                    curs.execute(query)
                    rows = curs.fetchall()
                    return (rows[0][0])
```

In [13]:
```
flow_detector_id=1295
start_time = '2016-07-31'
end_time = '2016-08-28'
mode = 'bicycle'
city='San Diego'
est_aadt = calculate_est_aadt(flow_detector_id, start_time,
                              end_time, city, mode)
print('Estimated AADt = {0}'.format(est_aadt))
```

```
Estimated AADt = 230.42
```