In [1]:

```
# Editor: Tearsyu
# PDS
# Generate history records
# Estimate number : 123774/356 = 347
import pandas as pd
import datetime
import random
import math

# Constraints: - 3000 clients (how many active clients?) and 173 parkings
#              - Paris accepts largest numbers of visiters in July, like a norma
l distribution
#              - At a common day, 00-06 clock there is less use times, from 7 to
 8, 12 to 13, 19 to 20, there is more use times
#              - We need a bundle of high random data with the constraints abov
e.


clients = 3000
parkings = 173
vehicles = parkings * 11
# weathers is a bulk of 2017 weather data of Paris
weathers = pd.read_csv("export-paris2017.csv", sep=',')
# getOneDayData(day, base_time_day) takes 2 arguments
# %day% the datetime like 2018-09-10
# %base_time_day% is the basic times of one day, according to our vision doc, th
is value is 347
def getOneDayData(day, base_time_day):
    currHours = day.hour
    currMonth = day.month
    # seedSign is used to add a random seed of times, some days there is more us
e times while some days less.
    seedSign = random.choice(["+", "-"]);
    hourTimes = 0
    res = []
    histId = 1
    precip_stat, visib_stat = getWeatherSeed(day)
    seedBaseHourTimes = int(base_time_day/24)
    for currHours in range(0, 24):
        if seedSign == '+':
            # baseHourTimes gives a basic use times of one hour, it ranges from
 347/12-squart(347/12) to 347/12+squart(347/12)
            baseHourTimes = seedBaseHourTimes + random.randint(0, seedBaseHourTi
mes + int(math.sqrt(seedBaseHourTimes)))
        else:
            baseHourTimes = seedBaseHourTimes - random.randint(0, seedBaseHourTi
mes - int(math.sqrt(seedBaseHourTimes)))

        # taux is an other random seed of ever hour, from 0.1 to 0.3 or from 1.4
 to 3.5,
        # this data comes from google map, the metro use times in every hour
        if (currHours >= 0 and currHours <= 6):
            taux = random.uniform(0.1, 0.3)
            hourTimes = int(baseHourTimes * taux)

        elif(currHours in [7, 8, 12, 13, 19, 20]):
            taux = random.uniform(1.2, 3.1)
            hourTimes = baseHourTimes + int(taux * baseHourTimes)
        else:
            hourTimes = baseHourTimes

        #print("hourTimes is " + str(hourTimes) + " at currHours is " + str(curr
Hours))

        for i in range(0, hourTimes):
            userId = random.randint(1, clients)
            depId = random.randint(1, parkings)
            arrId = random.choice([i for i in range(1, parkings) if i not in [de
pId]])
            vehicleId = random.randint(1, vehicles)
            depDateTime, arrDateTime = getTravelTime(day, currHours)
            lateMin = getLateMin(depDateTime, arrDateTime,  precip_stat, visib_s
tat)
            travelTime = arrDateTime - depDateTime
            basePrice, supPrice = getPrice(lateMin, travelTime)
            oneTube = [histId, userId, depId, arrId, vehicleId, depDateTime, arr
DateTime, lateMin, basePrice, supPrice]
            res.append(oneTube)
            histId = histId + 1
    #print("res length : " + str(len(res)) + " index is " + str(histId))
    return res
# generate a random traval's start time and it's duration
def getTravelTime(day, currHours):
    depMin = random.randint(0, 59)
    duration = random.randint(1, 60)
    depDateTime = day + datetime.timedelta(hours=currHours, minutes = depMin)
    arrDateTime = depDateTime + datetime.timedelta(minutes = duration)
    return depDateTime, arrDateTime;

# generate if a travel is late, the possibility is 85%(maybe here need to correc
t..)
# I add some factories to influence the late possibility, if it's july or augus
t, if it's in 8-9h, 18-19h
# there may be some transport problem..
def getLateMin(depDateTime, arrDateTime, precip_stat, visib_stat):
    seed = random.randint(1, 1000)
    seed = seed + precip_stat + visib_stat
    if depDateTime.month in [7, 8]:
        seed = seed + 80
    if depDateTime.hour in [8, 9, 18, 19]:
        seed = seed + 80
    if seed > 990 :
        minutes = arrDateTime - depDateTime;
        minutes = int((minutes.total_seconds()/60)*seed/1000)
        #print(minutes)
        if minutes == 0:
            minutes = 2
        return random.randint(1, minutes)
    else:
        return 0
# calculate a price for a trip
def getPrice(lateMin, travelTime):
    travelTime = int(travelTime.total_seconds()/60)
    seedSign = random.choice(["+", "-"])
    if seedSign is "+" :
        basePrice = travelTime * 1.00 - random.uniform(0.3, 1.2) * travelTime /
4
    else :
        basePrice = travelTime * 1.00 - random.uniform(0.3, 1.2) * travelTime /
2
```

```
    if lateMin == 0:
        return round(basePrice, 2), 0;
    else :
        return round(basePrice, 2), lateMin/2;

# This function take current date time as argument, and return a possibility of
 late
# A late travel depend on PRECIP_TOTAL_DAY_MM and VISIBILITY_AVG_KM
# for ex: if PRECIP_TOTAL_DAY_MM is greater than 2mm, this travel has more 30% p
ossibility to be late and go on
def getWeatherSeed(currDate):
    currDate = datetime.datetime.strftime(currDate,'%Y-%m-%d')
    getLine = weathers.loc[weathers['DATE'] == currDate]
    precipitation = float(getLine['PRECIP_TOTAL_DAY_MM'])
    visibility = float(getLine['VISIBILITY_AVG_KM'])
    precip_stat = 0
    visib_stat =0
    #print(str(precipitation) + " | " + str(visibility))
    if precipitation > 0.2 and precipitation < 1.0:
        precip_stat = 50
    elif precipitation >= 1.0 and precipitation < 3.0:
        precip_stat = 100
    elif precipitation >= 3.0 and precipitation < 5.0:
        precip_stat = 220
    elif precipitation >= 5.0:
        precip_stat = 350

    if visibility < 8.0 and visibility > 6.0:
        visib_stat = 50
    elif visibility <= 6.0 and visibility > 5.0:
        visib_stat = 100
    elif visibility <= 5.0 :
        visib_stat = 170
    return precip_stat, visib_stat



def generateYearHistory(base_time_day, year):
    cols = ["id", "client_id", "departure_id", "arrival_id","vehicle_id", "dep_t
ime", "arr_time",
            "late_time", "base_price", "sup_price"]

    oneYearHist = []
    oneDayHist = []
    currMon = 1
    currDate = 1
    currDay = datetime.datetime(year, currMon, currDate)
    seedSign = random.choice(["+", "-"]);

    for i in range(0, 356):
        #print("now at " + str(currDay) + " month is " + str(currDay.month))
        if currDay.month in [7, 8]:
            currBaseTimeDay = base_time_day + int(random.uniform(2, 4) * math.sq
rt(base_time_day))
        else:
            if seedSign is "+":
                currBaseTimeDay = base_time_day + int(random.uniform(0.3, 1.5) *
 math.sqrt(base_time_day))
            else :
                currBaseTimeDay = base_time_day - int(random.uniform(3, 8) * mat
h.sqrt(base_time_day))
```

```
        #print("base time of a day is " + str(currBaseTimeDay))
        oneDayHist = getOneDayData(currDay, currBaseTimeDay)
        oneYearHist = [*oneYearHist, *oneDayHist]
        currDay = currDay + datetime.timedelta(days = 1)
    #print("one year history length : " + str(len(oneYearHist)))
    df = pd.DataFrame(oneYearHist, columns=cols)
    return df

def testOnedayTimes(times):
    res = []
    for i in range(0, times):
        res.append(getOneDayData(347, 2017))
    print(res)

yearHistory = generateYearHistory(347, 2017)
#yearHistory.loc[yearHistory['late_time'] != 0]
yearHistory.to_csv("yearHistoryTemplate.csv", index=False, encoding='utf8')
yearHistory
```

```
Out[1]:
```

| | id | client_id | departure_id | arrival_id | vehicle_id | dep_time | arr_time | late_ti |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1730 | 82 | 71 | 1772 | 2017-01-01 00:15:00 | 2017-01-01 00:19:00 | 0 |
| **1** | 2 | 260 | 26 | 76 | 1441 | 2017-01-01 00:23:00 | 2017-01-01 00:27:00 | 0 |
| **2** | 3 | 726 | 150 | 91 | 38 | 2017-01-01 02:04:00 | 2017-01-01 02:31:00 | 0 |
| **3** | 4 | 1913 | 162 | 172 | 1791 | 2017-01-01 02:04:00 | 2017-01-01 02:18:00 | 0 |
| **4** | 5 | 1994 | 38 | 41 | 1874 | 2017-01-01 03:18:00 | 2017-01-01 04:17:00 | 0 |
| **5** | 6 | 791 | 145 | 25 | 307 | 2017-01-01 04:08:00 | 2017-01-01 05:07:00 | 0 |
| **6** | 7 | 1110 | 57 | 135 | 1504 | 2017-01-01 05:38:00 | 2017-01-01 06:02:00 | 0 |
| **7** | 8 | 484 | 41 | 7 | 524 | 2017-01-01 05:30:00 | 2017-01-01 06:30:00 | 0 |
| **8** | 9 | 1737 | 54 | 38 | 1082 | 2017-01-01 05:26:00 | 2017-01-01 06:16:00 | 0 |
| **9** | 10 | 1379 | 82 | 129 | 1067 | 2017-01-01 06:48:00 | 2017-01-01 06:57:00 | 0 |
| **10** | 11 | 234 | 144 | 95 | 273 | 2017-01-01 07:37:00 | 2017-01-01 07:41:00 | 0 |
| **11** | 12 | 1061 | 91 | 87 | 1378 | 2017-01-01 07:18:00 | 2017-01-01 07:20:00 | 0 |
| **12** | 13 | 229 | 60 | 51 | 1580 | 2017-01-01 07:43:00 | 2017-01-01 08:04:00 | 0 |
| **13** | 14 | 2164 | 18 | 63 | 606 | 2017-01-01 07:43:00 | 2017-01-01 08:05:00 | 0 |

| | id | client_id | departure_id | arrival_id | vehicle_id | dep_time | arr_time | late_ti |
|---|---|---|---|---|---|---|---|---|
| **14** | 15 | 2472 | 65 | 22 | 435 | 2017-01-01 07:34:00 | 2017-01-01 08:18:00 | 0 |
| **15** | 16 | 1047 | 166 | 135 | 1517 | 2017-01-01 07:58:00 | 2017-01-01 08:38:00 | 0 |
| **16** | 17 | 876 | 126 | 67 | 801 | 2017-01-01 07:47:00 | 2017-01-01 07:59:00 | 0 |
| **17** | 18 | 198 | 145 | 22 | 274 | 2017-01-01 07:05:00 | 2017-01-01 07:38:00 | 0 |
| **18** | 19 | 1874 | 37 | 31 | 1124 | 2017-01-01 07:38:00 | 2017-01-01 08:03:00 | 0 |
| **19** | 20 | 2729 | 131 | 162 | 1247 | 2017-01-01 07:40:00 | 2017-01-01 08:39:00 | 0 |
| **20** | 21 | 2288 | 133 | 49 | 400 | 2017-01-01 07:26:00 | 2017-01-01 07:43:00 | 0 |
| **21** | 22 | 1895 | 74 | 144 | 926 | 2017-01-01 07:34:00 | 2017-01-01 08:10:00 | 0 |
| **22** | 23 | 2514 | 50 | 43 | 256 | 2017-01-01 07:40:00 | 2017-01-01 08:05:00 | 0 |
| **23** | 24 | 2026 | 70 | 96 | 1753 | 2017-01-01 07:38:00 | 2017-01-01 07:55:00 | 0 |
| **24** | 25 | 1038 | 84 | 119 | 1043 | 2017-01-01 07:07:00 | 2017-01-01 07:31:00 | 0 |
| **25** | 26 | 89 | 144 | 93 | 1386 | 2017-01-01 07:55:00 | 2017-01-01 08:47:00 | 0 |
| **26** | 27 | 2524 | 84 | 16 | 363 | 2017-01-01 08:39:00 | 2017-01-01 09:08:00 | 0 |
| **27** | 28 | 1778 | 88 | 125 | 982 | 2017-01-01 08:08:00 | 2017-01-01 08:52:00 | 0 |

| | id | client_id | departure_id | arrival_id | vehicle_id | dep_time | arr_time | late_ti |
|---|---|---|---|---|---|---|---|---|
| **28** | 29 | 2174 | 101 | 36 | 851 | 2017-01-01 08:28:00 | 2017-01-01 08:50:00 | 0 |
| **29** | 30 | 1782 | 120 | 112 | 4 | 2017-01-01 08:23:00 | 2017-01-01 08:33:00 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **177570** | 713 | 1533 | 152 | 8 | 764 | 2017-12-22 22:32:00 | 2017-12-22 22:51:00 | 2 |
| **177571** | 714 | 2541 | 63 | 102 | 308 | 2017-12-22 22:07:00 | 2017-12-22 22:24:00 | 0 |
| **177572** | 715 | 1845 | 51 | 158 | 568 | 2017-12-22 22:29:00 | 2017-12-22 22:47:00 | 22 |
| **177573** | 716 | 1249 | 46 | 160 | 1248 | 2017-12-22 22:37:00 | 2017-12-22 22:51:00 | 0 |
| **177574** | 717 | 1711 | 78 | 165 | 881 | 2017-12-22 23:04:00 | 2017-12-22 23:51:00 | 0 |
| **177575** | 718 | 1730 | 165 | 162 | 1082 | 2017-12-22 23:31:00 | 2017-12-22 23:49:00 | 9 |
| **177576** | 719 | 109 | 34 | 3 | 510 | 2017-12-22 23:36:00 | 2017-12-23 00:28:00 | 0 |
| **177577** | 720 | 2288 | 169 | 111 | 917 | 2017-12-22 23:37:00 | 2017-12-23 00:25:00 | 0 |
| **177578** | 721 | 1930 | 111 | 82 | 1629 | 2017-12-22 23:30:00 | 2017-12-22 23:56:00 | 0 |
| **177579** | 722 | 2297 | 55 | 4 | 9 | 2017-12-22 23:16:00 | 2017-12-22 23:36:00 | 0 |
| **177580** | 723 | 2363 | 103 | 44 | 671 | 2017-12-22 23:35:00 | 2017-12-23 00:27:00 | 0 |
| **177581** | 724 | 2657 | 112 | 163 | 1003 | 2017-12-22 23:29:00 | 2017-12-23 00:09:00 | 21 |

| | id | client_id | departure_id | arrival_id | vehicle_id | dep_time | arr_time | late_ti |
|---|---|---|---|---|---|---|---|---|
| **177582** | 725 | 2478 | 82 | 164 | 1581 | 2017-12-22 23:39:00 | 2017-12-22 23:59:00 | 0 |
| **177583** | 726 | 178 | 160 | 36 | 252 | 2017-12-22 23:14:00 | 2017-12-22 23:51:00 | 6 |
| **177584** | 727 | 203 | 85 | 117 | 1191 | 2017-12-22 23:18:00 | 2017-12-22 23:55:00 | 0 |
| **177585** | 728 | 505 | 85 | 14 | 1569 | 2017-12-22 23:24:00 | 2017-12-22 23:39:00 | 0 |
| **177586** | 729 | 328 | 57 | 165 | 105 | 2017-12-22 23:43:00 | 2017-12-23 00:07:00 | 19 |
| **177587** | 730 | 2065 | 22 | 121 | 454 | 2017-12-22 23:22:00 | 2017-12-23 00:03:00 | 0 |
| **177588** | 731 | 92 | 153 | 111 | 623 | 2017-12-22 23:37:00 | 2017-12-23 00:03:00 | 27 |
| **177589** | 732 | 1030 | 8 | 106 | 495 | 2017-12-22 23:07:00 | 2017-12-22 23:56:00 | 31 |
| **177590** | 733 | 887 | 145 | 72 | 773 | 2017-12-22 23:48:00 | 2017-12-23 00:13:00 | 0 |
| **177591** | 734 | 2542 | 40 | 15 | 864 | 2017-12-22 23:12:00 | 2017-12-22 23:37:00 | 1 |
| **177592** | 735 | 2277 | 149 | 101 | 1470 | 2017-12-22 23:48:00 | 2017-12-23 00:22:00 | 0 |
| **177593** | 736 | 1312 | 140 | 156 | 801 | 2017-12-22 23:49:00 | 2017-12-23 00:08:00 | 0 |
| **177594** | 737 | 2727 | 20 | 146 | 629 | 2017-12-22 23:48:00 | 2017-12-23 00:14:00 | 0 |
| **177595** | 738 | 2596 | 35 | 46 | 1477 | 2017-12-22 23:01:00 | 2017-12-22 23:19:00 | 0 |

| | id | client_id | departure_id | arrival_id | vehicle_id | dep_time | arr_time | late_ti |
|---|---|---|---|---|---|---|---|---|
| **177596** | 739 | 741 | 129 | 146 | 1276 | 2017-12-22 23:24:00 | 2017-12-22 23:56:00 | 15 |
| **177597** | 740 | 1806 | 153 | 63 | 62 | 2017-12-22 23:44:00 | 2017-12-22 23:50:00 | 0 |
| **177598** | 741 | 2645 | 63 | 84 | 1369 | 2017-12-22 23:47:00 | 2017-12-23 00:03:00 | 12 |
| **177599** | 742 | 1331 | 64 | 45 | 1148 | 2017-12-22 23:58:00 | 2017-12-23 00:32:00 | 8 |

177600 rows × 10 columns

In [2]:

```
# Editor: Tearsyu
# PDS
# Generate records of booking
# Constraints: - BIMyCar accepts bookings in 10 days
#              - Is it running in real time?

import datetime
import pandas as pd
import random
import math

clients = 3000
parkings = 173

def generateOneDayBookings(startTime, currBaseTimeDay):
    currHours = startTime.hour
    currMonth = startTime.month
    # seedSign is used to add a random seed of times, some days there is more us
e times while some days less.
    seedSign = random.choice(["+", "-"]);
    hourTimes = 0
    res = []
    histId = 1
    seedBaseHourTimes = int(currBaseTimeDay/24)
    for currHours in range(currHours, 24):
        if seedSign == '+':
            # baseHourTimes gives a basic use times of one hour, it ranges from
 347/12-squart(347/12) to 347/12+squart(347/12)
            baseHourTimes = seedBaseHourTimes + random.randint(0, seedBaseHourTi
mes + int(math.sqrt(seedBaseHourTimes)))
        else:
            baseHourTimes = seedBaseHourTimes - random.randint(0, seedBaseHourTi
mes - int(math.sqrt(seedBaseHourTimes)))

        # taux is an other random seed of ever hour, from 0.1 to 0.3 or from 1.4
 to 3.5,
        # this data comes from google map, the metro use times in every hour
        if (currHours >= 0 and currHours <= 6):
            taux = random.uniform(0.1, 0.3)
            hourTimes = int(baseHourTimes * taux)

        elif(currHours in [7, 8, 12, 13, 19, 20]):
            taux = random.uniform(1.4, 3.5)
            hourTimes = baseHourTimes + int(taux * baseHourTimes)
        else:
            hourTimes = baseHourTimes

        for i in range(0, hourTimes):
            userId = random.randint(1, clients)
            depId = random.randint(1, parkings)
            arrId = random.choice([i for i in range(1, parkings) if i not in [de
pId]])
            depDateTime, arrDateTime = getTravelTime(startTime, currHours)
            depDateTime = depDateTime.replace(second = 0, microsecond=0)
            arrDateTime = arrDateTime.replace(second = 0, microsecond=0)
            travelTime = arrDateTime - depDateTime
            basePrice = getPriceBooking(travelTime)
            oneTube = [histId, userId, depId, arrId, depDateTime, arrDateTime, b
asePrice]
```

```
            res.append(oneTube)
            histId = histId + 1
    #print("res length : " + str(len(res)) + " index is " + str(histId))
    return res

#Need correct
def getTravelTime(startTime, currHours):
    depMin = random.randint(startTime.minute, 59)
    duration = random.randint(1, 60)
    newDay = datetime.datetime(startTime.year, startTime.month, startTime.day)
    depDateTime = newDay + datetime.timedelta(hours=currHours, minutes = depMin)
    arrDateTime = depDateTime + datetime.timedelta(minutes = duration)
    return depDateTime, arrDateTime;

def getPriceBooking(travelTime):
    travelTime = int(travelTime.total_seconds()/60)
    seedSign = random.choice(["+", "-"])
    if seedSign is "+" :
        basePrice = travelTime * 0.70 - random.uniform(0.3, 1.2) * travelTime /
4
    else :
        basePrice = travelTime * 0.70 - random.uniform(0.3, 1.2) * travelTime /
2
    if basePrice < 2:
        basePrice = 2
    return round(basePrice, 2)

def generateBookings(startTime, baseDayTimes):
    cols = ["id", "client_id", "departure_id", "arrival_id", "dep_time", "estima
te_arr_time", "price"]
    oneDayHist = []
    allBookings = []
    seedSign = random.choice(["+", "-"])
    currDay = startTime
    for i in range(0, 10):
        if currDay.month in [7, 8]:
            currBaseTimeDay = baseDayTimes + int(random.uniform(2, 4) * math.sqr
t(baseDayTimes))
        else:
            if seedSign is "+":
                currBaseTimeDay = baseDayTimes + int(random.uniform(0.3, 1.5) *
math.sqrt(baseDayTimes))
            else :
                currBaseTimeDay = baseDayTimes - int(random.uniform(3, 8) * math
.sqrt(baseDayTimes))
        #print("now " + str(currDay))
        oneDayHist = generateOneDayBookings(currDay, currBaseTimeDay)
        allBookings = [*allBookings, *oneDayHist]
        print(currDay)
        currDay = datetime.datetime(currDay.year, currDay.month, currDay.day) +
datetime.timedelta(days = 1)
    #print("one year history length : " + str(len(oneYearHist)))
    df = pd.DataFrame(allBookings, columns=cols)
    return df

startTime = datetime.datetime.now()
bookings = generateBookings(startTime, 347)
#bookings.to_csv("bookings.csv", index=False, encoding='utf8')
bookings
```

```
2018-12-20 11:20:26.076784
2018-12-21 00:00:00
2018-12-22 00:00:00
2018-12-23 00:00:00
2018-12-24 00:00:00
2018-12-25 00:00:00
2018-12-26 00:00:00
2018-12-27 00:00:00
2018-12-28 00:00:00
2018-12-29 00:00:00
```

```
Out[2]:
```

| | id | client_id | departure_id | arrival_id | dep_time | estimate_arr_time | price |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 893 | 117 | 84 | 2018-12-20 11:32:00 | 2018-12-20 12:10:00 | 16.34 |
| **1** | 2 | 2041 | 108 | 10 | 2018-12-20 11:20:00 | 2018-12-20 11:46:00 | 3.98 |
| **2** | 3 | 35 | 15 | 45 | 2018-12-20 11:43:00 | 2018-12-20 12:20:00 | 13.61 |
| **3** | 4 | 815 | 123 | 131 | 2018-12-20 11:54:00 | 2018-12-20 12:43:00 | 20.27 |
| **4** | 5 | 451 | 56 | 60 | 2018-12-20 11:47:00 | 2018-12-20 12:22:00 | 13.82 |
| **5** | 6 | 2099 | 94 | 51 | 2018-12-20 11:41:00 | 2018-12-20 12:03:00 | 11.38 |
| **6** | 7 | 1323 | 80 | 153 | 2018-12-20 11:57:00 | 2018-12-20 12:21:00 | 11.50 |
| **7** | 8 | 1901 | 125 | 139 | 2018-12-20 12:32:00 | 2018-12-20 13:07:00 | 8.66 |
| **8** | 9 | 1364 | 171 | 78 | 2018-12-20 12:48:00 | 2018-12-20 12:51:00 | 2.00 |
| **9** | 10 | 2768 | 165 | 46 | 2018-12-20 12:52:00 | 2018-12-20 13:37:00 | 15.77 |
| **10** | 11 | 377 | 129 | 156 | 2018-12-20 12:49:00 | 2018-12-20 13:30:00 | 20.45 |
| **11** | 12 | 2985 | 64 | 163 | 2018-12-20 12:36:00 | 2018-12-20 12:50:00 | 7.51 |
| **12** | 13 | 1020 | 137 | 47 | 2018-12-20 12:51:00 | 2018-12-20 12:54:00 | 2.00 |
| **13** | 14 | 1263 | 86 | 143 | 2018-12-20 12:36:00 | 2018-12-20 12:56:00 | 2.23 |
| **14** | 15 | 164 | 103 | 115 | 2018-12-20 12:37:00 | 2018-12-20 13:22:00 | 27.85 |
| **15** | 16 | 824 | 118 | 68 | 2018-12-20 12:25:00 | 2018-12-20 12:27:00 | 2.00 |
| **16** | 17 | 386 | 153 | 105 | 2018-12-20 12:30:00 | 2018-12-20 13:17:00 | 5.65 |
| **17** | 18 | 2919 | 63 | 85 | 2018-12-20 12:52:00 | 2018-12-20 13:34:00 | 21.90 |
| **18** | 19 | 240 | 166 | 169 | 2018-12-20 12:44:00 | 2018-12-20 13:09:00 | 10.66 |

| | id | client_id | departure_id | arrival_id | dep_time | estimate_arr_time | price |
|---|---|---|---|---|---|---|---|
| **19** | 20 | 1419 | 35 | 117 | 2018-12-20 12:39:00 | 2018-12-20 13:25:00 | 26.80 |
| **20** | 21 | 1558 | 42 | 122 | 2018-12-20 12:59:00 | 2018-12-20 13:13:00 | 5.72 |
| **21** | 22 | 1399 | 140 | 110 | 2018-12-20 12:31:00 | 2018-12-20 13:31:00 | 19.10 |
| **22** | 23 | 910 | 97 | 69 | 2018-12-20 12:34:00 | 2018-12-20 12:57:00 | 11.26 |
| **23** | 24 | 890 | 85 | 119 | 2018-12-20 12:50:00 | 2018-12-20 13:39:00 | 14.81 |
| **24** | 25 | 1787 | 66 | 111 | 2018-12-20 12:25:00 | 2018-12-20 13:07:00 | 11.05 |
| **25** | 26 | 1126 | 93 | 51 | 2018-12-20 12:54:00 | 2018-12-20 12:56:00 | 2.00 |
| **26** | 27 | 760 | 88 | 96 | 2018-12-20 12:24:00 | 2018-12-20 12:31:00 | 4.29 |
| **27** | 28 | 334 | 31 | 116 | 2018-12-20 12:21:00 | 2018-12-20 12:53:00 | 13.69 |
| **28** | 29 | 417 | 117 | 55 | 2018-12-20 12:42:00 | 2018-12-20 12:46:00 | 2.11 |
| **29** | 30 | 2655 | 24 | 152 | 2018-12-20 12:25:00 | 2018-12-20 12:26:00 | 2.00 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **4785** | 742 | 1268 | 27 | 42 | 2018-12-29 22:25:00 | 2018-12-29 22:57:00 | 19.92 |
| **4786** | 743 | 1534 | 106 | 16 | 2018-12-29 22:39:00 | 2018-12-29 22:41:00 | 2.00 |
| **4787** | 744 | 2551 | 101 | 5 | 2018-12-29 22:49:00 | 2018-12-29 23:04:00 | 7.21 |
| **4788** | 745 | 247 | 73 | 115 | 2018-12-29 22:21:00 | 2018-12-29 23:05:00 | 22.15 |
| **4789** | 746 | 2282 | 3 | 96 | 2018-12-29 22:39:00 | 2018-12-29 23:21:00 | 20.46 |
| **4790** | 747 | 1481 | 9 | 24 | 2018-12-29 22:40:00 | 2018-12-29 23:26:00 | 21.52 |
| **4791** | 748 | 1703 | 3 | 152 | 2018-12-29 22:10:00 | 2018-12-29 22:52:00 | 21.03 |
| **4792** | 749 | 2146 | 157 | 151 | 2018-12-29 22:00:00 | 2018-12-29 22:02:00 | 2.00 |
| **4793** | 750 | 61 | 41 | 5 | 2018-12-29 23:53:00 | 2018-12-30 00:05:00 | 4.95 |

| | id | client_id | departure_id | arrival_id | dep_time | estimate_arr_time | price |
|---|---|---|---|---|---|---|---|
| **4794** | 751 | 1812 | 59 | 140 | 2018-12-29 23:10:00 | 2018-12-29 23:14:00 | 2.00 |
| **4795** | 752 | 1133 | 39 | 74 | 2018-12-29 23:45:00 | 2018-12-30 00:05:00 | 4.02 |
| **4796** | 753 | 2486 | 51 | 29 | 2018-12-29 23:36:00 | 2018-12-29 23:52:00 | 8.28 |
| **4797** | 754 | 1126 | 50 | 145 | 2018-12-29 23:18:00 | 2018-12-30 00:12:00 | 30.70 |
| **4798** | 755 | 2016 | 19 | 73 | 2018-12-29 23:03:00 | 2018-12-29 23:24:00 | 10.20 |
| **4799** | 756 | 1524 | 80 | 136 | 2018-12-29 23:39:00 | 2018-12-29 23:57:00 | 6.80 |
| **4800** | 757 | 1496 | 173 | 13 | 2018-12-29 23:22:00 | 2018-12-29 23:41:00 | 7.74 |
| **4801** | 758 | 617 | 23 | 44 | 2018-12-29 23:46:00 | 2018-12-30 00:46:00 | 29.15 |
| **4802** | 759 | 2724 | 138 | 114 | 2018-12-29 23:23:00 | 2018-12-29 23:24:00 | 2.00 |
| **4803** | 760 | 12 | 162 | 112 | 2018-12-29 23:04:00 | 2018-12-29 23:14:00 | 4.23 |
| **4804** | 761 | 1493 | 80 | 57 | 2018-12-29 23:10:00 | 2018-12-29 23:54:00 | 23.59 |
| **4805** | 762 | 1640 | 11 | 65 | 2018-12-29 23:22:00 | 2018-12-30 00:21:00 | 16.11 |
| **4806** | 763 | 637 | 172 | 45 | 2018-12-29 23:47:00 | 2018-12-30 00:20:00 | 20.21 |
| **4807** | 764 | 474 | 151 | 77 | 2018-12-29 23:17:00 | 2018-12-29 23:57:00 | 19.65 |
| **4808** | 765 | 221 | 127 | 65 | 2018-12-29 23:22:00 | 2018-12-29 23:45:00 | 14.11 |
| **4809** | 766 | 695 | 49 | 52 | 2018-12-29 23:48:00 | 2018-12-30 00:05:00 | 8.13 |
| **4810** | 767 | 704 | 53 | 63 | 2018-12-29 23:47:00 | 2018-12-30 00:27:00 | 18.93 |
| **4811** | 768 | 2393 | 135 | 115 | 2018-12-29 23:46:00 | 2018-12-29 23:49:00 | 2.00 |
| **4812** | 769 | 670 | 105 | 36 | 2018-12-29 23:07:00 | 2018-12-29 23:17:00 | 2.00 |
| **4813** | 770 | 2126 | 132 | 110 | 2018-12-29 23:56:00 | 2018-12-30 00:39:00 | 16.51 |

| | id | client_id | departure_id | arrival_id | dep_time | estimate_arr_time | price |
|---|---|---|---|---|---|---|---|
| **4814** | 771 | 2475 | 113 | 37 | 2018-12-29 23:49:00 | 2018-12-30 00:28:00 | 19.15 |

4815 rows × 7 columns

```
In [138]:
```

```
# This function take current date time as argument, and return a possibility of
 late
# A late travel depend on PRECIP_TOTAL_DAY_MM and VISIBILITY_AVG_KM
# for ex: if PRECIP_TOTAL_DAY_MM is greater than 2mm, this travel has more 30% p
ossibility to be late and go on
def getWeatherSeed(currDate):
    currDate = datetime.datetime.strftime(currDate,'%Y-%m-%d')
    getLine = weathers.loc[weathers['DATE'] == currDate]
    precipitation = float(getLine['PRECIP_TOTAL_DAY_MM'])
    visibility = float(getLine['VISIBILITY_AVG_KM'])
    precip_stat = 0
    visib_stat =0
    print(str(precipitation) + " | " + str(visibility))
    if precipitation > 0.2 and precipitation < 1.0:
        precip_stat = 200
    elif precipitation >= 1.0 and precipitation < 5.0 :
        precip_stat = 300
    elif precipitation >= 5.0:
        precip_stat = 550

    if visibility < 8.0 and visibility > 6.0:
        visib_stat = 50
    elif visibility <= 6.0 and visibility > 5.0:
        visib_stat = 100
    elif visibility <= 5.0 :
        visib_stat = 170
    return precip_stat, visib_stat

getWeatherSeed(datetime.datetime(2017, 10, 2))
```

```
2.3 | 9.75
```

```
Out[138]:
```

```
(300, 0)
```

In [6]:

```
# Table du meteo
weathers.head(5)
```

Out[6]:

| | DATE | MAX_TEMPERATURE_C | MIN_TEMPERATURE_C | WINDSPEED_MAX_KMH |
|---|---|---|---|---|
| **0** | 2017-01-01 | 3 | -1 | 14 |
| **1** | 2017-01-02 | 4 | 2 | 10 |
| **2** | 2017-01-03 | 5 | 1 | 9 |
| **3** | 2017-01-04 | 6 | 3 | 18 |
| **4** | 2017-01-05 | 6 | 1 | 13 |

In [7]:

```
# Table parking

parking = pd.read_csv("new_parking.csv")
parking.head(5)
```

Out[7]:

| | Unnamed: 0 | ID | NOM_PARC | ADRESS_GEO | Arrdt | TEL | geo_point_2d |
|---|---|---|---|---|---|---|---|
| **0** | 161 | 1 | AMPERE | 93 TER RUE AMPERE | 17 | 01 43 80 73 81 | 48.885238365, 2.29863751292 |
| **1** | 169 | 2 | MALESHERBES ANJOU | 20 TER BOULEVARD MALESHERBES | 8 | ND | 48.8724867829, 2.32181194243 |
| **2** | 132 | 3 | ECOLE DE MEDECINE | 8 TER RUE DE L ECOLE DE MEDECINE | 6 | 01 43 29 61 38 | 48.8509460984, 2.34112737464 |
| **3** | 13 | 4 | VILLIERS | 14 AVENUE DE VILLIERS | 17 | 01 47 63 44 91 | 48.8819558206, 2.3142426079 |
| **4** | 111 | 6 | VENDOME | 26 TER PLACE VENDOME | 1 | 01 42 60 50 00 | 48.86789756, 2.33019790757 |