# COMP60711 coursework 3

Language and tool selection:
Python: Task5.1, Task5.2, Task6, Task7.1
Excel: Task7.2

# TASK 5.1

## (1) Output:

Two numerical values, one for each of the two columns:

```
The column completeness of Gap is:
98.03654443753885
The column completeness of Headway is:
98.96532007458049
```

Figure 1.output

## (2) Intermediate steps

Step 1. Get data frame with Tuesdays between 7:00 and 19:00 condition.

```
1  import numpy as np
2  import datetime
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  import dateutil.parser
6
7  list1 = pd.read_csv("rawpvr_2018-02-01_28d_1083 TueFri.csv")
8  list2 = pd.read_csv("rawpvr_2018-02-01_28d_1083 TueFri.csv")['Date']
9  Week_of_day =[]
10 for date in list2:
11     formed_date = dateutil.parser.parse(date)
12     Week_of_day.append(formed_date.weekday())
13
14 list1['week of day'] = Week_of_day
15
16 task5_1 = list1.loc[(list1['week of day'] == 1)]
17
18 Date_information = pd.to_datetime(task5_1.Date)
19 Date_with_day = Date_information.dt.floor('D')
20 Date_with_hour = Date_information - Date_with_day
21 task51_7_19 = Date_with_hour.between(pd.Timedelta('07:00:00'), pd.Timedelta('19:00:00'),inclusive="left")
22 task51_7_19_df = task5_1.loc[task51_7_19]
23 print(task51_7_19_df)
```

Figure 2.Code

To filter with Tuesday condition, it adds one more column called “week of day”.  In line 10-12, it creates a new list which use weekday function to return the week of day value.
In this case, the value returned from Monday to Sunday is 0 to 6. Therefore, Tuesday is 1. Line 16 filter with this condition to get data frame with Tuesday only.
The next problem is time range, for this task. We are not interested in the year, month, day data from the date frame. We just need to focus on the time range between 7:00 and 19:00
From lines 18 to 22, its purpose is to get the time range from 7:00 - 19:00. Due to the date value is not uniform for this data frame, line 18 uses the to_datetime function to convert the argument to datetime. All of these data changes to the year-month-day-hour-minute-second-fs format. Then line 19 gets the date with only

the year-month-day format. Line 20, uses minus to get the “Date_with_hour” variable with data only time range. Finally, use between functions to get time range from 7:00-19:00. Then select this time range in the data frame.
Here is the data frame with time range from 7:00 and 19:00 Tuesday only:

| | Date | Lane | Lane Name | Direction | ... | Gap (s) | Flags | Flag Text | week of day |
|---|---|---|---|---|---|---|---|---|---|
| 68516 | 2018-02-06 07:00:00.100000 | 3 | NB_OS | 1 | ... | NaN | 0 | NaN | 1 |
| 68517 | 2018-02-06 07:00:01.160000 | 2 | NB_MID | 1 | ... | NaN | 0 | NaN | 1 |
| 68518 | 2018-02-06 07:00:03 | 2 | NB_MID | 1 | ... | 1.992 | 0 | NaN | 1 |
| 68519 | 2018-02-06 07:00:03.020000 | 4 | SB_OS | 2 | ... | NaN | 0 | NaN | 1 |
| 68520 | 2018-02-06 07:00:03.030000 | 1 | NB_NS | 1 | ... | NaN | 0 | NaN | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 496058 | 2018-02-27 18:59:56.010000 | 6 | SB_NS | 2 | ... | 3.001 | 0 | NaN | 1 |
| 496059 | 2018-02-27 18:59:57.050000 | 6 | SB_NS | 2 | ... | 1.127 | 0 | NaN | 1 |
| 496060 | 2018-02-27 18:59:58.040000 | 1 | NB_NS | 1 | ... | 5.182 | 0 | NaN | 1 |
| 496061 | 2018-02-27 18:59:58.060000 | 2 | NB_MID | 1 | ... | 7.307 | 0 | NaN | 1 |
| 496062 | 2018-02-27 18:59:59.090000 | 6 | SB_NS | 2 | ... | 1.133 | 0 | NaN | 1 |

Figure 3.Output with week of day

## Step 2. Calculate column completeness of “Gap” and “Headway”

```
25  task5_gap = task51_7_19_df['Gap (s)']
26  empty_gap = task5_gap.isna().sum()
27  gap_Completeness = ((len(task5_gap) - empty_gap)*100)/len(task5_gap)
28  print("The column completeness of Gap is:")
29  print(gap_Completeness)
30
31  task5_headway = task51_7_19_df['Headway (s)']
32  empty_headway = task5_headway.isna().sum()
33  headway_Completeness = ((len(task5_headway) - empty_headway)*100)/len(task5_headway)
34  print("The column completeness of Headway is:")
35  print(headway_Completeness)
```

Figure 4.Code2

To get this value, we need to consider the given formula:
Column_Completeness = (number_of_non-empty_cells x 100) / number_of_cells
In this case, we create a new data frame with column “Gap” only. Then use isna() function to count the number of empty cells for the column gap. After that, use the total number of gap cells minus empty cells to get non-empty cells and times 100 to get the percentage. Finally, divide by the total number of cells. This value is the column completeness of the gap, and we could print it out.
Line 31-35 is the “Headway” version with the same process
Output:

```
The column completeness of Gap is:
98.03654443753885
The column completeness of Headway is:
98.96532007458049
```

Figure 5. Output task5.1

# (3) Interpretation of the result

In conclusion, these incomplete data affect the data quality in this case. It may contain multiple reasons with human or collection problems. However, considering the dataset is huge. Moreover, the data is generated in multiple rows in a second. These effects increase the difficulty to collect complete data. At the same time, the column completeness of gap and headway is close to 98%. It makes the data

quality is fairly great, though there are some missing values in the gap and headway column.

# TASK 5.2

## (1)Output:

```
The median of gap from 7:00-8:00 :
1.996
The median of headway from 7:00-8:00 :
2.752
                            Date Lane Name  Headway (s)  Gap (s)
68517   2018-02-06 07:00:01.160000    NB_MID        1.720    1.996
68518          2018-02-06 07:00:03    NB_MID        2.480    1.992
68521   2018-02-06 07:00:04.020000    NB_MID        1.520    0.856
68524   2018-02-06 07:00:05.020000    NB_MID        1.226    0.680
68529   2018-02-06 07:00:07.020000    NB_MID        2.137    1.694
...                            ...       ...          ...      ...
453758  2018-02-27 07:59:45.070000    NB_MID        4.629    1.452
453761         2018-02-27 07:59:48    NB_MID        3.800    1.757
453767  2018-02-27 07:59:51.010000    NB_MID        3.046    2.180
453770  2018-02-27 07:59:54.020000    NB_MID        2.653    2.297
453774         2018-02-27 07:59:57    NB_MID        2.775    2.156
```

```
The median of gap from 8:00-9:00 :
2.101
The median of headway from 8:00-9:00 :
3.0
                                Date Lane Name  Headway (s)  Gap (s)
73854           2018-02-06 08:00:02    NB_MID        3.267    2.101
73859    2018-02-06 08:00:05.080000    NB_MID        3.528    2.253
73863    2018-02-06 08:00:07.080000    NB_MID        2.492    1.366
73865    2018-02-06 08:00:10.030000    NB_MID        4.015    2.905
73869    2018-02-06 08:00:13.030000    NB_MID        3.333    2.363
...                              ...       ...          ...      ...
458345   2018-02-27 08:59:50.060000    NB_MID        2.448    0.960
458347   2018-02-27 08:59:52.010000    NB_MID        3.287    1.578
458351          2018-02-27 08:59:54    NB_MID        2.146    1.227
458354   2018-02-27 08:59:57.030000    NB_MID        3.960    2.691
458357          2018-02-27 08:59:59    NB_MID        2.057    1.066
```

```
The median of gap from 9:00-10:00 :
2.3085
The median of headway from 9:00-10:00 :
2.917
                                Date Lane Name  Headway (s)  Gap (s)
78847   2018-02-06 09:00:01.030000    NB_MID         5.026   2.3085
78852   2018-02-06 09:00:04.080000    NB_MID         3.000   2.2220
78857   2018-02-06 09:00:08.070000    NB_MID         4.605   3.5770
78860   2018-02-06 09:00:09.020000    NB_MID         1.718   1.1570
78862   2018-02-06 09:00:11.030000    NB_MID         2.529   1.6990
...                             ...       ...           ...      ...
462156          2018-02-27 09:59:41    NB_MID         2.008   1.3410
462163          2018-02-27 09:59:49    NB_MID         8.000   7.6880
462170  2018-02-27 09:59:54.050000    NB_MID         5.500   5.2710
462174  2018-02-27 09:59:56.080000    NB_MID         1.523   1.0020
462175  2018-02-27 09:59:57.010000    NB_MID         1.350   1.0020
```

```
The median of gap from 10:00-11:00 :
3.112
The median of headway from 10:00-11:00 :
3.521
                                Date Lane Name  Headway (s)  Gap (s)
82946   2018-02-06 10:00:07.030000    NB_MID         3.521   3.112
82950   2018-02-06 10:00:08.040000    NB_MID         1.212   0.644
82951   2018-02-06 10:00:09.030000    NB_MID         1.240   0.591
82953          2018-02-06 10:00:11    NB_MID         1.996   1.316
82956   2018-02-06 10:00:13.040000    NB_MID         2.319   2.040
...                             ...       ...           ...      ...
465109  2018-02-27 10:59:11.020000    NB_MID        13.800  13.254
465112  2018-02-27 10:59:14.060000    NB_MID         2.463   2.080
465126  2018-02-27 10:59:27.050000    NB_MID        13.900  13.641
465151  2018-02-27 10:59:54.090000    NB_MID         1.861  26.182
465154  2018-02-27 10:59:55.050000    NB_MID         1.737   1.271
```

```
The median of gap from 11:00-12:00 :
3.153
The median of headway from 11:00-12:00 :
3.544
                                Date Lane Name  Headway (s)  Gap (s)
86181   2018-02-06 11:00:12.040000    NB_MID         3.544   3.153
86182          2018-02-06 11:00:14    NB_MID         1.831   1.353
86186   2018-02-06 11:00:15.220000    NB_MID         1.410   0.944
86201   2018-02-06 11:00:33.020000    NB_MID         2.241   3.153
86202   2018-02-06 11:00:35.060000    NB_MID         1.446   1.069
...                             ...       ...           ...      ...
468080  2018-02-27 11:59:28.090000    NB_MID         2.100   1.766
468087  2018-02-27 11:59:45.090000    NB_MID        17.000  16.730
468089  2018-02-27 11:59:47.030000    NB_MID         2.441   2.124
468095          2018-02-27 11:59:54    NB_MID         1.604   6.279
468097  2018-02-27 11:59:55.010000    NB_MID         1.514   0.803
```

```
The median of gap from 12:00-13:00 :
2.964
The median of headway from 12:00-13:00 :
3.356
                              Date Lane Name  Headway (s)  Gap (s)
89354   2018-02-06 12:00:00.110000    NB_MID        4.075    2.964
89359   2018-02-06 12:00:06.020000    NB_MID        7.486    5.822
89372   2018-02-06 12:00:19.090000    NB_MID        3.356    2.964
89375          2018-02-06 12:00:21    NB_MID        2.368    1.868
89378   2018-02-06 12:00:23.070000    NB_MID        1.800    1.466
...                            ...       ...          ...      ...
471219         2018-02-27 12:59:22    NB_MID        5.469    5.663
471223  2018-02-27 12:59:25.030000    NB_MID        3.787    3.023
471227  2018-02-27 12:59:29.040000    NB_MID        3.450    3.770
471238  2018-02-27 12:59:43.080000    NB_MID       13.400   13.130
471240  2018-02-27 12:59:44.050000    NB_MID        1.871    1.386

[2411 rows x 4 columns]
```

```
The median of gap from 13:00-14:00 :
2.882
The median of headway from 13:00-14:00 :
3.254
                              Date Lane Name  Headway (s)  Gap (s)
92784   2018-02-06 13:00:01.170000    NB_MID        1.627    2.882
92787   2018-02-06 13:00:04.030000    NB_MID        1.096    0.384
92789   2018-02-06 13:00:04.060000    NB_MID        3.233    2.533
92790   2018-02-06 13:00:06.090000    NB_MID        1.690    1.279
92793   2018-02-06 13:00:10.040000    NB_MID        4.415    4.221
...                            ...       ...          ...      ...
474341  2018-02-27 13:58:18.070000    NB_MID       20.400   19.779
474352  2018-02-27 13:58:27.080000    NB_MID        9.100    8.772
474354  2018-02-27 13:58:30.040000    NB_MID        4.018    3.246
474364  2018-02-27 13:58:42.010000    NB_MID       11.700   11.211
474391  2018-02-27 13:59:03.010000    NB_MID        4.680   19.201

[2386 rows x 4 columns]
```

```
The median of gap from 14:00-15:00 :
2.6725000000000003
The median of headway from 14:00-15:00 :
3.075
                              Date Lane Name  Headway (s)  Gap (s)
96443   2018-02-06 14:00:00.060000    NB_MID        1.547   1.0410
96450   2018-02-06 14:00:16.190000    NB_MID        3.075   2.6725
96451   2018-02-06 14:00:18.030000    NB_MID        2.865   2.1220
96454   2018-02-06 14:00:20.060000    NB_MID        1.333   0.9690
96458   2018-02-06 14:00:24.010000    NB_MID        4.881   4.0930
...                            ...       ...          ...      ...
477787         2018-02-27 14:59:07    NB_MID       45.000  44.6240
477788  2018-02-27 14:59:09.020000    NB_MID        1.941   1.9040
477805  2018-02-27 14:59:26.020000    NB_MID       17.000  16.7640
477808  2018-02-27 14:59:28.080000    NB_MID        1.766   1.2950
477809  2018-02-27 14:59:29.020000    NB_MID        1.596   1.1220

[2243 rows x 4 columns]
```

```
The median of gap from 15:00-16:00 :
2.109
The median of headway from 15:00-16:00 :
2.604
                                Date Lane Name  Headway (s)  Gap (s)
100091  2018-02-06 15:00:02.030000    NB_MID        2.604    2.109
100105  2018-02-06 15:00:25.040000    NB_MID        2.604    2.109
100117  2018-02-06 15:00:42.050000    NB_MID       17.100   16.754
100124  2018-02-06 15:00:46.060000    NB_MID        3.416    2.816
100125  2018-02-06 15:00:47.040000    NB_MID        1.984    0.573
...                             ...       ...          ...      ...
481797  2018-02-27 15:59:29.020000    NB_MID        4.144    6.295
481805  2018-02-27 15:59:35.020000    NB_MID        6.000    5.807
481809  2018-02-27 15:59:39.030000    NB_MID        4.547    3.830
481821  2018-02-27 15:59:51.050000    NB_MID       12.200   11.922
481828  2018-02-27 15:59:59.090000    NB_MID        7.400    7.102

[2246 rows x 4 columns]
```

```
The median of gap from 16:00-17:00 :
1.8105
The median of headway from 16:00-17:00 :
2.45
                               Date Lane Name  Headway (s)  Gap (s)
104585  2018-02-06 16:00:01.190000    NB_MID        2.450   1.8105
104589  2018-02-06 16:00:04.080000    NB_MID        4.696   2.5880
104597  2018-02-06 16:00:08.040000    NB_MID        4.711   4.2480
104600  2018-02-06 16:00:10.070000    NB_MID        1.358   1.1160
104657  2018-02-06 16:01:17.060000    NB_MID        2.450   1.8105
...                             ...       ...          ...      ...
486883  2018-02-27 16:59:49.030000    NB_MID        3.064   2.3230
486889  2018-02-27 16:59:52.080000    NB_MID        2.479   2.1710
486894  2018-02-27 16:59:54.060000    NB_MID        2.094   1.3920
486896  2018-02-27 16:59:55.030000    NB_MID        1.987   1.4130
486902  2018-02-27 16:59:58.090000    NB_MID        2.925   2.2770

[2886 rows x 4 columns]
```

```
The median of gap from 17:00-18:00 :
1.877
The median of headway from 17:00-18:00 :
2.584
                               Date Lane Name  Headway (s)  Gap (s)
110218  2018-02-06 17:00:00.010000    NB_MID        1.500    1.877
110224  2018-02-06 17:00:03.010000    NB_MID        3.600    2.640
110234  2018-02-06 17:00:07.030000    NB_MID        2.432    3.760
110241  2018-02-06 17:00:11.050000    NB_MID        5.580    3.752
110251  2018-02-06 17:00:17.020000    NB_MID        4.962    1.877
...                             ...       ...          ...      ...
492080  2018-02-27 17:59:20.070000    NB_MID        2.063    1.571
492090         2018-02-27 17:59:26    NB_MID        2.864    6.000
492094  2018-02-27 17:59:28.010000    NB_MID        2.336    1.756
492099  2018-02-27 17:59:29.050000    NB_MID        1.760    1.071
492109  2018-02-27 17:59:36.060000    NB_MID        6.423    5.780

[3422 rows x 4 columns]
```

```
The median of gap from 18:00-19:00 :
2.04
The median of headway from 18:00-19:00 :
2.636
                              Date Lane Name  Headway (s)  Gap (s)
115619  2018-02-06 18:00:01.190000    NB_MID        2.636    2.040
115621         2018-02-06 18:00:04    NB_MID        5.100    2.865
115629  2018-02-06 18:00:06.090000    NB_MID        2.146    1.600
115632  2018-02-06 18:00:07.040000    NB_MID        1.765    1.237
115639  2018-02-06 18:00:11.040000    NB_MID        4.612    3.702
...                            ...       ...          ...      ...
496010  2018-02-27 18:58:51.060000    NB_MID        1.200    0.725
496029  2018-02-27 18:59:26.080000    NB_MID       35.200   34.941
496050  2018-02-27 18:59:48.040000    NB_MID       22.600   22.382
496051         2018-02-27 18:59:50    NB_MID        1.800    1.343
496061  2018-02-27 18:59:58.060000    NB_MID        7.600    7.307

[2770 rows x 4 columns]
```

Figure 6. Output task5.2

# (2) Intermediate steps

```
1  import numpy as np
2  import datetime
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  import dateutil.parser
6
7
8  list1 = pd.read_csv("rawpvr_2018-02-01_28d_1083 TueFri.csv")
9  list2 = pd.read_csv("rawpvr_2018-02-01_28d_1083 TueFri.csv")['Date']
10 Week_of_day =[]
11 for date in list2:
12     formed_date = dateutil.parser.parse(date)
13     Week_of_day.append(formed_date.weekday())
14
15 list1['week of day'] = Week_of_day
16
17 Tuesday_North = list1.loc[(list1['week of day'] == 1)&(list1['Lane Name']== "NB_MID")]
18 print(Tuesday_North)
19 North_list =[]
20 Date_information = pd.to_datetime(Tuesday_North.Date)
21 Date_with_day = Date_information.dt.floor('D')
22 Date_with_hour = Date_information - Date_with_day
23 North_7_8 = Date_with_hour.between(pd.Timedelta('07:00:00'), pd.Timedelta('08:00:00'),inclusive="left")
24 test1 = Tuesday_North.loc[North_7_8]
```

Figure 7.code

These steps are similar to task 5.1, create a column for week of day filter and choose a time range without year-month-day datetime. line 24 returns a new data frame with the appropriate time range with the “NB_MID” lane and Tuesday.

```
26  Tuesday_7_helper = list1.loc[list1['week of day'] == 1]
27  Date_information1 = pd.to_datetime(Tuesday_7_helper.Date)
28  Date_with_day1 = Date_information1.dt.floor('D')
29  Date_with_hour1 = Date_information1 - Date_with_day1
30  Tuesday_7_8 =Date_with_hour1.between(pd.Timedelta('07:00:00'), pd.Timedelta('08:00:00'),inclusive="left")
31  test1_gaps = test1[['Date','Lane Name','Headway (s)','Gap (s)']]
32  test2 = Tuesday_7_helper.loc[Tuesday_7_8]
33  test1_gaps['Gap (s)'] = test1_gaps['Gap (s)'].fillna(test2['Gap (s)'].median())
34  test1_gaps['Headway (s)'] = test1_gaps['Headway (s)'].fillna(test2['Headway (s)'].median())
35  print("The median of gap from 7:00-8:00 :")
36  print(test2['Gap (s)'].median())
37  print("The median of headway from 7:00-8:00 :")
38  print(test2['Headway (s)'].median())
39  print(test1_gaps)
```

Figure 8.code

Consider the example provided in the task, it said
“For example, if missing values are found on Tuesday 06/02/2018 - 10:00 and Tuesday 20/02/2018 - 15:00, then you should calculate the median of gap (or headway) considering all Tuesdays at 10:00 and all Tuesdays at 15:00 to obtain two values,”
Therefore, in the calculation, we should not only focus on the NB_MID lane in this case. line 26 to 30 create another new data frame with Tuesday only for calculating the median.

Due to I have added an additional column called “week of day” in the data frame. It makes the printed data frame is not able to show column” Headway”. Therefore, another new data frame was created in line 31. This data frame only contains information we need in this sub-task.

| | Date | Lane | Lane Name | Direction | ... | Gap (s) | Flags | Flag Text | week of day |
|---|---|---|---|---|---|---|---|---|---|
| 64603 | 2018-02-06 00:00:14.020000 | 2 | NB_MID | 1 | ... | NaN | 0 | NaN | 1 |
| 64604 | 2018-02-06 00:00:41.060000 | 2 | NB_MID | 1 | ... | NaN | 0 | NaN | 1 |
| 64608 | 2018-02-06 00:01:24.050000 | 2 | NB_MID | 1 | ... | 43.600 | 0 | NaN | 1 |
| 64610 | 2018-02-06 00:01:29.040000 | 2 | NB_MID | 1 | ... | 4.630 | 0 | NaN | 1 |
| 64616 | 2018-02-06 00:01:54.030000 | 2 | NB_MID | 1 | ... | 24.633 | 0 | NaN | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 503754 | 2018-02-27 23:57:44.020000 | 2 | NB_MID | 1 | ... | 85.088 | 0 | NaN | 1 |
| 503756 | 2018-02-27 23:58:00.050000 | 2 | NB_MID | 1 | ... | 15.982 | 0 | NaN | 1 |
| 503761 | 2018-02-27 23:58:53.050000 | 2 | NB_MID | 1 | ... | 51.725 | 0 | NaN | 1 |
| 503762 | 2018-02-27 23:58:56.080000 | 2 | NB_MID | 1 | ... | 3.016 | 0 | NaN | 1 |
| 503763 | 2018-02-27 23:59:00.090000 | 2 | NB_MID | 1 | ... | 3.833 | 0 | NaN | 1 |

Figure 9.Output

Line 33 and 34 use fillna function to fill in the null value in the data frame. The median function is used to compute the median values.
Finally, the line 35 to 39 print the missing value with associated day times and update the dataset.
Then we could change the time range value in line 30 manually to calculate and fill in the null value with another time range.

# TASK 6

## (1) Output:

```
C:\Users\38139\COMP60711\task6>python3 Task6.py
6.292904217952499
6.006590110356325
7.276777086305146
6.668723580953999
7.51633842121931
The values from these five north lanes are:
[6.292904217952499, 6.006590110356325, 7.276777086305146, 6.668723580953999, 7.51633842121931]
The average value is:
6.7522666833574565
```

Figure 10.Output task6

## (2) Intermediate steps

Line 1 to 24 are similar to before, create “week of day” column to filter with Friday condition. Choose lane name as “NB_MID” and find the required time range 17:00 - 18:00.

```
1  import numpy as np
2  import datetime
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  import dateutil.parser
6
7  list_result = []
8
9  list1 = pd.read_csv("rawpvr_2018-02-01_28d_1083 TueFri.csv")
10 list2 = pd.read_csv("rawpvr_2018-02-01_28d_1083 TueFri.csv")['Date']
11 Week_of_day =[]
12 for date in list2:
13     formed_date = dateutil.parser.parse(date)
14     Week_of_day.append(formed_date.weekday())
15
16 list1['week of day'] = Week_of_day
17
18 site1083_MID = list1.loc[(list1['week of day'] == 4)&(list1['Lane Name']== "NB_MID")]
19
20 Date_information = pd.to_datetime(site1083_MID.Date)
21 Date_with_day = Date_information.dt.floor('D')
22 Date_with_hour = Date_information - Date_with_day
23 mid1083 = Date_with_hour.between(pd.Timedelta('17:00:00'), pd.Timedelta('18:00:00'),inclusive="left")
24 mid1083_17_18 = site1083_MID.loc[mid1083]
```

Figure 11.Code

Then get the new data frame with speed only. Then use the mean function to calculate the average speed. After that, we need to be careful with the unit. Due to the task is given 4.86km and the unit of speed is mph. We need to convert mph to kph. From some basic searching [1], I have found 1mph = 1.6 kph. Therefore, mph times 1.61 to convert to kph in line 27. Follow the journey time calculation in the task requirement in line 28. Then print it out and add it to the list. We will use this list to get the average value later.

```
mid1083_speed = mid1083_17_18['Speed (mph)']
average_mid_speed = mid1083_speed.mean(axis=0)
mid_kph = average_mid_speed*1.61
mid_result = (4.86*60)/mid_kph
print(mid_result)
list_result.append(mid_result)
```

Figure 12.Code

Output for the average journey time in NB_MID is :

```
6.292904217952499
```

Figure 13.Output

Then we use the same codes and ideas for the other 2 lanes in site 1083.

```
site1083_OS = list1.loc[(list1['week of day'] == 4)&(list1['Lane Name']== "NB_OS")]

Date_information_OS = pd.to_datetime(site1083_OS.Date)
Date_with_day_OS = Date_information_OS.dt.floor('D')
Date_with_hour_OS = Date_information_OS - Date_with_day_OS
os1083 = Date_with_hour_OS.between(pd.Timedelta('17:00:00'), pd.Timedelta('18:00:00'),inclusive="left")
os1083_17_18 = site1083_OS.loc[os1083]
os1083_speed = os1083_17_18['Speed (mph)']
average_os_speed = os1083_speed.mean(axis=0)
os_kph = average_os_speed*1.61
os_result = (4.86*60)/os_kph
print(os_result)
list_result.append(os_result)

site1083_NS = list1.loc[(list1['week of day'] == 4)&(list1['Lane Name']== "NB_NS")]

Date_information_NS = pd.to_datetime(site1083_NS.Date)
Date_with_day_NS = Date_information_NS.dt.floor('D')
Date_with_hour_NS = Date_information_NS - Date_with_day_NS
ns1083 = Date_with_hour_NS.between(pd.Timedelta('17:00:00'), pd.Timedelta('18:00:00'),inclusive="left")
ns1083_17_18 = site1083_NS.loc[ns1083]
ns1083_speed = ns1083_17_18['Speed (mph)']
average_ns_speed = ns1083_speed.mean(axis=0)
ns_kph = average_ns_speed*1.61
ns_result = (4.86*60)/ns_kph
print(ns_result)
list_result.append(ns_result)
```

Figure 14.Code

Output for the average journey time in NB_OS is :

```
6.006590110356325
```

Figure 15.Output NB_OS

Output for the average journey time in NB_NS is :

```
7.276777086305146
```

Figure 16.Output NB_NS

After that, processing site 1415 data with the same steps.

```
list3 = pd.read_csv("rawpvr_2018-02-01_28d_1415 TueFri.csv")
list4 = pd.read_csv("rawpvr_2018-02-01_28d_1415 TueFri.csv")['Date']
Week_of_day1 =[]
for date in list4:
    formed_date = dateutil.parser.parse(date)
    Week_of_day1.append(formed_date.weekday())

list3['week of day'] = Week_of_day1

site1415_ne = list3.loc[(list3['week of day'] == 4)&(list3['Lane Name']== "NE_OS")]

Date_information_NE = pd.to_datetime(site1415_ne.Date)
Date_with_day_NE = Date_information_NE.dt.floor('D')
Date_with_hour_NE = Date_information_NE - Date_with_day_NE
ne1415 = Date_with_hour_NE.between(pd.Timedelta('17:00:00'), pd.Timedelta('18:00:00'),inclusive="left")
ne1415_17_18 = site1415_ne.loc[ne1415]
ne1415_speed = ne1415_17_18['Speed (mph)']
average_ne_speed = ne1415_speed.mean(axis=0)
ne_kph = average_ne_speed*1.61
ne_result = (4.86*60)/ne_kph
print(ne_result)
list_result.append(ne_result)
```

```
site1415_nens = list3.loc[(list3['week of day'] == 4)&(list3['Lane Name']== "NE_NS")]

Date_information_NENS = pd.to_datetime(site1415_nens.Date)
Date_with_day_NENS = Date_information_NENS.dt.floor('D')
Date_with_hour_NENS = Date_information_NENS - Date_with_day_NENS
nens1415 = Date_with_hour_NENS.between(pd.Timedelta('17:00:00'), pd.Timedelta('18:00:00'),inclusive="left")
nens1415_17_18 = site1415_nens.loc[nens1415]
nens1415_speed = nens1415_17_18['Speed (mph)']
average_nens_speed = nens1415_speed.mean(axis=0)
nens_kph = average_nens_speed*1.61
nens_result = (4.86*60)/nens_kph
print(nens_result)
list_result.append(nens_result)
```

Figure 17.Code

Finally, print the list of these five values and get the average value.

```
print("The values from these five north lanes are:")
print(list_result)
print("The average value is:")
average_value = sum(list_result)/len(list_result)
print(average_value)
```

Figure 18.Print Code

Output:

```
C:\Users\38139\COMP60711\task6>python3 Task6.py
6.292904217952499
6.006590110356325
7.276777086305146
6.668723580953999
7.51633842121931
The values from these five north lanes are:
[6.292904217952499, 6.006590110356325, 7.276777086305146, 6.668723580953999, 7.51633842121931]
The average value is:
6.7522666833574565
```

Figure 19.Output task6

# (3) Interpretation of the results

These five values are very close. For north lane data, the average Friday journey time from sites 1415 is higher than sites 1083. At the same time,

the missing value of column speed is significantly less than column gap and headway. It means this set of data quality is fairly worth further research.

# TASK 7.1

## (i)

The Record completeness is a possible solution to measure row completeness. The Record completeness allows the user to get a basic understanding of how data complete [2]. It will compare the length of the input document to expectations length to check whether input length matches the defined range or expectations length [2]. Therefore, it is a sensible approach to measure row completeness in this case.

There are three attributes which needed to measure row completeness, “Record Completeness”, “Complete Attributes” and “Matching Records” [2]. The “Complete Attributes” means how much un-empty data are in the data frame. The “Record Completeness” is a percentage value for how many un-null values are for current rows. The formula would be :

Record Completeness = (number_of_non-empty_cells x 100) / number_of_cells for each row.

The “Matching Records” means how many rows are matched the current Record Completeness condition.

## (ii)

The purpose of this part is to try to get these three attributes for each row: “Record Completeness”, “Complete Attributes” and “Matching Records”. These three attributes have been discussed in 7.1 (i) part before.

Step 1. Get data with associate with Tuesdays between 7:00 and 19:00

```python
import numpy as np
import datetime
import matplotlib.pyplot as plt
import pandas as pd
import dateutil.parser


list1 = pd.read_csv("rawpvr_2018-02-01_28d_1083 TueFri.csv")
list2 = pd.read_csv("rawpvr_2018-02-01_28d_1083 TueFri.csv")['Date']
Week_of_day =[]
for date in list2:
    formed_date = dateutil.parser.parse(date)
    Week_of_day.append(formed_date.weekday())

list1['week of day'] = Week_of_day

task7_1 = list1.loc[(list1['week of day'] == 1)]

Date_information = pd.to_datetime(task7_1.Date)
Date_with_day = Date_information.dt.floor('D')
Date_with_hour = Date_information - Date_with_day
task7_7_19 = Date_with_hour.between(pd.Timedelta('07:00:00'), pd.Timedelta('19:00:00'),inclusive="left")
task7_7_19_df = task7_1.loc[task7_7_19]

print(task7_7_19_df)
```

Figure 20. Code

The first 27 line code is very similar to before. We collect all data with multiple conditions: Tuesday and  7:00 and 19:00.

| | Date | Lane | Lane Name | Direction | ... | Gap (s) | Flags | Flag Text | week of day |
|---|---|---|---|---|---|---|---|---|---|
| 68516 | 2018-02-06 07:00:00.100000 | 3 | NB_OS | 1 | ... | NaN | 0 | NaN | 1 |
| 68517 | 2018-02-06 07:00:01.160000 | 2 | NB_MID | 1 | ... | NaN | 0 | NaN | 1 |
| 68518 | 2018-02-06 07:00:03 | 2 | NB_MID | 1 | ... | 1.992 | 0 | NaN | 1 |
| 68519 | 2018-02-06 07:00:03.020000 | 4 | SB_OS | 2 | ... | NaN | 0 | NaN | 1 |
| 68520 | 2018-02-06 07:00:03.030000 | 1 | NB_NS | 1 | ... | NaN | 0 | NaN | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 496058 | 2018-02-27 18:59:56.010000 | 6 | SB_NS | 2 | ... | 3.001 | 0 | NaN | 1 |
| 496059 | 2018-02-27 18:59:57.050000 | 6 | SB_NS | 2 | ... | 1.127 | 0 | NaN | 1 |
| 496060 | 2018-02-27 18:59:58.040000 | 1 | NB_NS | 1 | ... | 5.182 | 0 | NaN | 1 |
| 496061 | 2018-02-27 18:59:58.060000 | 2 | NB_MID | 1 | ... | 7.307 | 0 | NaN | 1 |
| 496062 | 2018-02-27 18:59:59.090000 | 6 | SB_NS | 2 | ... | 1.133 | 0 | NaN | 1 |

Figure 21. Output with time range

The output is the data from 7:00 - 19:00 on Tuesday.

## Step 2. add “Complete Attribute” and “Record Completeness” to the data frame

```python
print(task7_7_19_df)
list_complete = task7_7_19_df.apply(lambda data_7_19_unempty: data_7_19_unempty.count()-1, axis=1)

task7_7_19_df['Complete Attribute'] = list_complete
record_list =[]
for data in list_complete:
    record_list.append((data*100)/10)
task7_7_19_df['Record Completeness'] = record_list

print(task7_7_19_df)
```

Figure 22. Code

line 26 creates a list.  This list contains the number of un-empty data for each row. Due to we have to build a column called “week of day” to filter with Tuesday condition before. At the same time, all rows contain this value and it would not be empty. We would not want to include this column for row completeness in this case. Therefore, we use “-1” to not count this column.

line 28 add “Complete Attribute” column, line 29-32 add “Record Completeness” column with a simple calculation. Since the original data contains 10 columns, we divide all un-empty value by 10.

| | Date | Lane | Lane Name | ... | week of day | Complete Attribute | Record Completeness |
|---|---|---|---|---|---|---|---|
| 68516 | 2018-02-06 07:00:00.100000 | 3 | NB_OS | ... | 1 | 8 | 80.0 |
| 68517 | 2018-02-06 07:00:01.160000 | 2 | NB_MID | ... | 1 | 8 | 80.0 |
| 68518 | 2018-02-06 07:00:03 | 2 | NB_MID | ... | 1 | 9 | 90.0 |
| 68519 | 2018-02-06 07:00:03.020000 | 4 | SB_OS | ... | 1 | 7 | 70.0 |
| 68520 | 2018-02-06 07:00:03.030000 | 1 | NB_NS | ... | 1 | 8 | 80.0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 496058 | 2018-02-27 18:59:56.010000 | 6 | SB_NS | ... | 1 | 9 | 90.0 |
| 496059 | 2018-02-27 18:59:57.050000 | 6 | SB_NS | ... | 1 | 9 | 90.0 |
| 496060 | 2018-02-27 18:59:58.040000 | 1 | NB_NS | ... | 1 | 9 | 90.0 |
| 496061 | 2018-02-27 18:59:58.060000 | 2 | NB_MID | ... | 1 | 9 | 90.0 |
| 496062 | 2018-02-27 18:59:59.090000 | 6 | SB_NS | ... | 1 | 9 | 90.0 |

Figure 23. Output with Record Completeness

Finally, we use these lines to get and output the “Matching Records” value for this data frame.

```
36  Match_records = task7_7_19_df.groupby('Record Completeness').count()['Complete Attribute']
37  print(Match_records)
```

Figure 24. Code with matching record

```
Record Completeness
70.0        2081
80.0        1877
90.0      197167
Name: Complete Attribute, dtype: int64
```

Figure 25. Output with matching record

## (iii)

Completeness is a way to measure the data quality of a data set. This formula is a possible way to measure the data quality in this case. However, we may not be able to say which formula is better. Since the formula, we used in task 5.1 is focused on the data missed for each column. This formula is measured data missed for each row. However, most of the rows missed data in columns “Flag Text”, “Gap” and “Headway”. Other columns almost do not have any missed data. Therefore, the formula we used in task 7.1 is not able to explain which columns lack data well. And the formula we used in task 5.1 is only able to measure the particular columns of data.

# TASK 7.2

## We use excel as technology to solve this task.

### Step 1. Create “week of day” column

Due to there does not have a week of day attribute and help us filter with Friday condition, we need to add one more column as the first step.

### Select “Insert” and “Entire column”

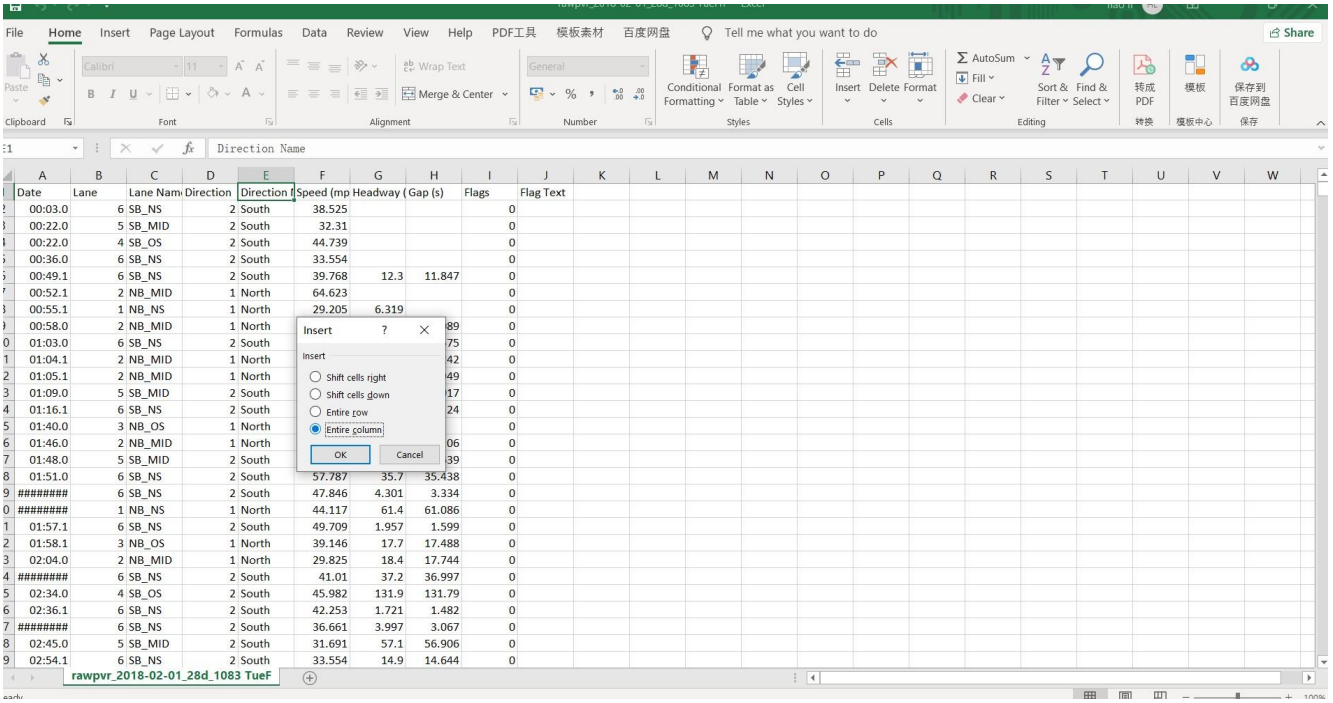
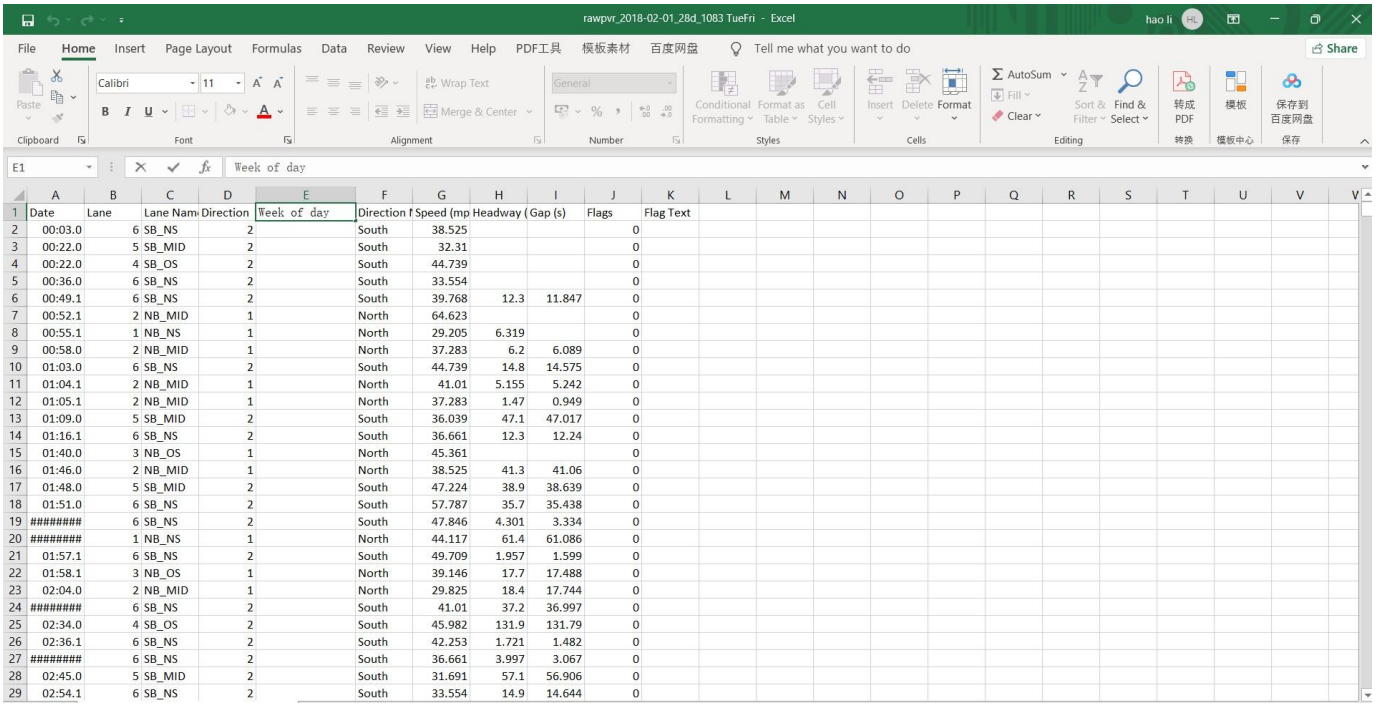



Figure 26. insert column

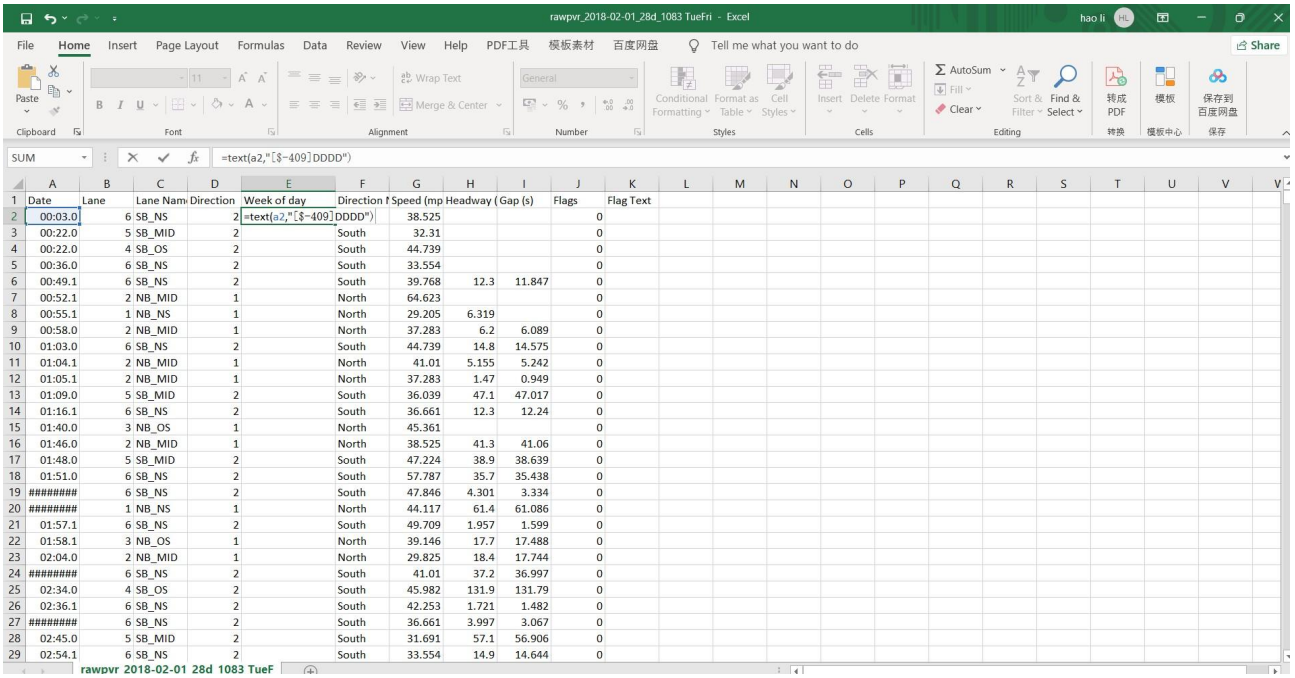
### Fill in the column name with “week of day”



| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Date | Lane | Lane Nam | Direction | Week of day | Direction | Speed (mp | Headway ( | Gap (s) | Flags | Flag Text |
| 2 | 00:03.0 | 6 | SB_NS | 2 | | South | 38.525 | | | 0 | |
| 3 | 00:22.0 | 5 | SB_MID | 2 | | South | 32.31 | | | 0 | |
| 4 | 00:22.0 | 4 | SB_OS | 2 | | South | 44.739 | | | 0 | |
| 5 | 00:36.0 | 6 | SB_NS | 2 | | South | 33.554 | | | 0 | |
| 6 | 00:49.1 | 6 | SB_NS | 2 | | South | 39.768 | 12.3 | 11.847 | 0 | |
| 7 | 00:52.1 | 2 | NB_MID | 1 | | North | 64.623 | | | 0 | |
| 8 | 00:55.1 | 1 | NB_NS | 1 | | North | 29.205 | 6.319 | | 0 | |
| 9 | 00:58.0 | 2 | NB_MID | 1 | | North | 37.283 | 6.2 | 6.089 | 0 | |
| 10 | 01:03.0 | 6 | SB_NS | 2 | | South | 44.739 | 14.8 | 14.575 | 0 | |
| 11 | 01:04.1 | 2 | NB_MID | 1 | | North | 41.01 | 5.155 | 5.242 | 0 | |
| 12 | 01:05.1 | 2 | NB_MID | 1 | | North | 37.283 | 1.47 | 0.949 | 0 | |
| 13 | 01:09.0 | 5 | SB_MID | 2 | | South | 36.039 | 47.1 | 47.017 | 0 | |
| 14 | 01:16.1 | 6 | SB_NS | 2 | | South | 36.661 | 12.3 | 12.24 | 0 | |
| 15 | 01:40.0 | 3 | NB_OS | 1 | | North | 45.361 | | | 0 | |
| 16 | 01:46.0 | 2 | NB_MID | 1 | | North | 38.525 | 41.3 | 41.06 | 0 | |
| 17 | 01:48.0 | 5 | SB_MID | 2 | | South | 47.224 | 38.9 | 38.639 | 0 | |
| 18 | 01:51.0 | 6 | SB_NS | 2 | | South | 57.787 | 35.7 | 35.438 | 0 | |
| 19 | ######## | 6 | SB_NS | 2 | | South | 47.846 | 4.301 | 3.334 | 0 | |
| 20 | ######## | 1 | NB_NS | 1 | | North | 44.117 | 61.4 | 61.086 | 0 | |
| 21 | 01:57.1 | 6 | SB_NS | 2 | | South | 49.709 | 1.957 | 1.599 | 0 | |
| 22 | 01:58.1 | 3 | NB_OS | 1 | | North | 39.146 | 17.7 | 17.488 | 0 | |
| 23 | 02:04.0 | 2 | NB_MID | 1 | | North | 29.825 | 18.4 | 17.744 | 0 | |
| 24 | ######## | 6 | SB_NS | 2 | | South | 41.01 | 37.2 | 36.997 | 0 | |
| 25 | 02:34.0 | 4 | SB_OS | 2 | | South | 45.982 | 131.9 | 131.79 | 0 | |
| 26 | 02:36.1 | 6 | SB_NS | 2 | | South | 42.253 | 1.721 | 1.482 | 0 | |
| 27 | ######## | 6 | SB_NS | 2 | | South | 36.661 | 3.997 | 3.067 | 0 | |
| 28 | 02:45.0 | 5 | SB_MID | 2 | | South | 31.691 | 57.1 | 56.906 | 0 | |
| 29 | 02:54.1 | 6 | SB_NS | 2 | | South | 33.554 | 14.9 | 14.644 | 0 | |

Figure 27. Column name

Use the “text” function to get the week of day information. [409] purposes to convert the "week of day" value from my mother language to English.



| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Date | Lane | Lane Nam | Direction | Week of day | Direction | Speed (mp | Headway ( | Gap (s) | Flags | Flag Text |
| 2 | 00:03.0 | 6 | SB_NS | 2 | =text(a2,"[$-409]DDDD") | South | 38.525 | | | 0 | |
| 3 | 00:22.0 | 5 | SB_MID | 2 | | South | 32.31 | | | 0 | |
| 4 | 00:22.0 | 4 | SB_OS | 2 | | South | 44.739 | | | 0 | |
| 5 | 00:36.0 | 6 | SB_NS | 2 | | South | 33.554 | | | 0 | |
| 6 | 00:49.1 | 6 | SB_NS | 2 | | South | 39.768 | 12.3 | 11.847 | 0 | |
| 7 | 00:52.1 | 2 | NB_MID | 1 | | North | 64.623 | | | 0 | |
| 8 | 00:55.1 | 1 | NB_NS | 1 | | North | 29.205 | 6.319 | | 0 | |
| 9 | 00:58.0 | 2 | NB_MID | 1 | | North | 37.283 | 6.2 | 6.089 | 0 | |
| 10 | 01:03.0 | 6 | SB_NS | 2 | | South | 44.739 | 14.8 | 14.575 | 0 | |
| 11 | 01:04.1 | 2 | NB_MID | 1 | | North | 41.01 | 5.155 | 5.242 | 0 | |
| 12 | 01:05.1 | 2 | NB_MID | 1 | | North | 37.283 | 1.47 | 0.949 | 0 | |
| 13 | 01:09.0 | 5 | SB_MID | 2 | | South | 36.039 | 47.1 | 47.017 | 0 | |
| 14 | 01:16.1 | 6 | SB_NS | 2 | | South | 36.661 | 12.3 | 12.24 | 0 | |
| 15 | 01:40.0 | 3 | NB_OS | 1 | | North | 45.361 | | | 0 | |
| 16 | 01:46.0 | 2 | NB_MID | 1 | | North | 38.525 | 41.3 | 41.06 | 0 | |
| 17 | 01:48.0 | 5 | SB_MID | 2 | | South | 47.224 | 38.9 | 38.639 | 0 | |
| 18 | 01:51.0 | 6 | SB_NS | 2 | | South | 57.787 | 35.7 | 35.438 | 0 | |
| 19 | ######## | 6 | SB_NS | 2 | | South | 47.846 | 4.301 | 3.334 | 0 | |
| 20 | ######## | 1 | NB_NS | 1 | | North | 44.117 | 61.4 | 61.086 | 0 | |
| 21 | 01:57.1 | 6 | SB_NS | 2 | | South | 49.709 | 1.957 | 1.599 | 0 | |
| 22 | 01:58.1 | 3 | NB_OS | 1 | | North | 39.146 | 17.7 | 17.488 | 0 | |
| 23 | 02:04.0 | 2 | NB_MID | 1 | | North | 29.825 | 18.4 | 17.744 | 0 | |
| 24 | ######## | 6 | SB_NS | 2 | | South | 41.01 | 37.2 | 36.997 | 0 | |
| 25 | 02:34.0 | 4 | SB_OS | 2 | | South | 45.982 | 131.9 | 131.79 | 0 | |
| 26 | 02:36.1 | 6 | SB_NS | 2 | | South | 42.253 | 1.721 | 1.482 | 0 | |
| 27 | ######## | 6 | SB_NS | 2 | | South | 36.661 | 3.997 | 3.067 | 0 | |
| 28 | 02:45.0 | 5 | SB_MID | 2 | | South | 31.691 | 57.1 | 56.906 | 0 | |
| 29 | 02:54.1 | 6 | SB_NS | 2 | | South | 33.554 | 14.9 | 14.644 | 0 | |

Figure 28. Function

Then click the right bottom corner to make other rows return this value

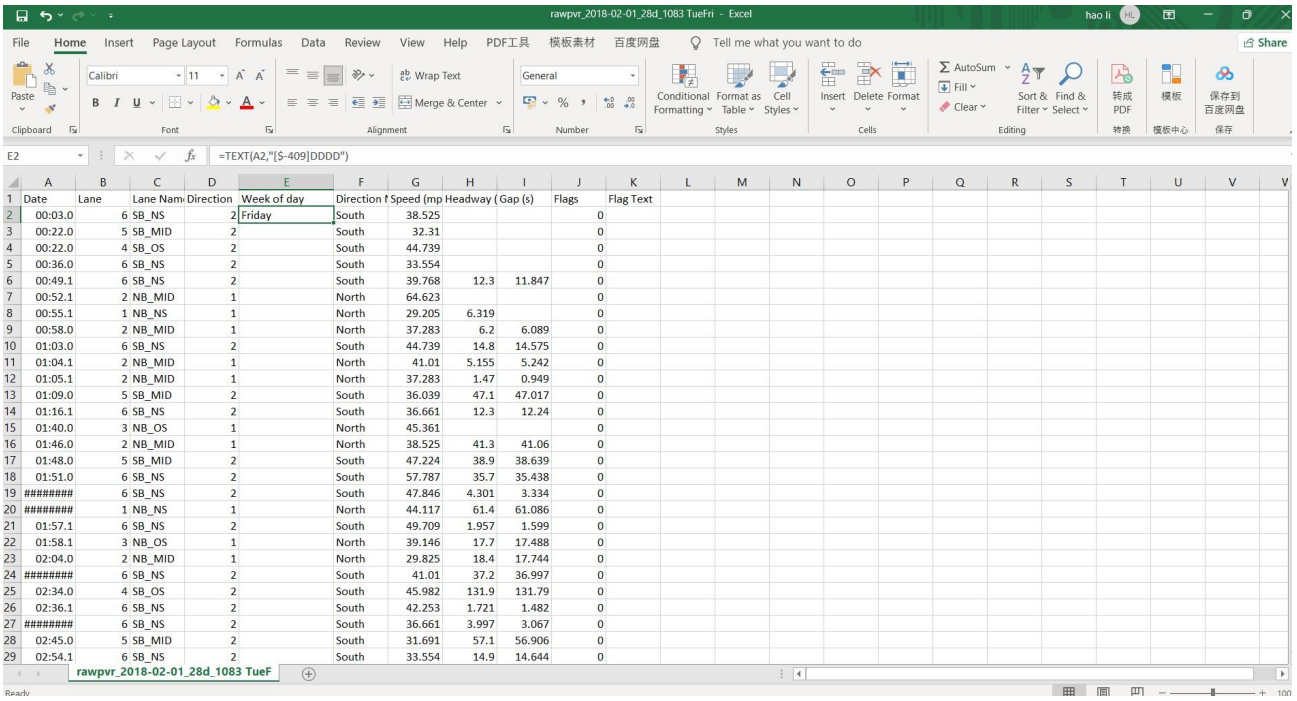



Figure 29. Function output

Then select “Insert” and choose “pivottable”

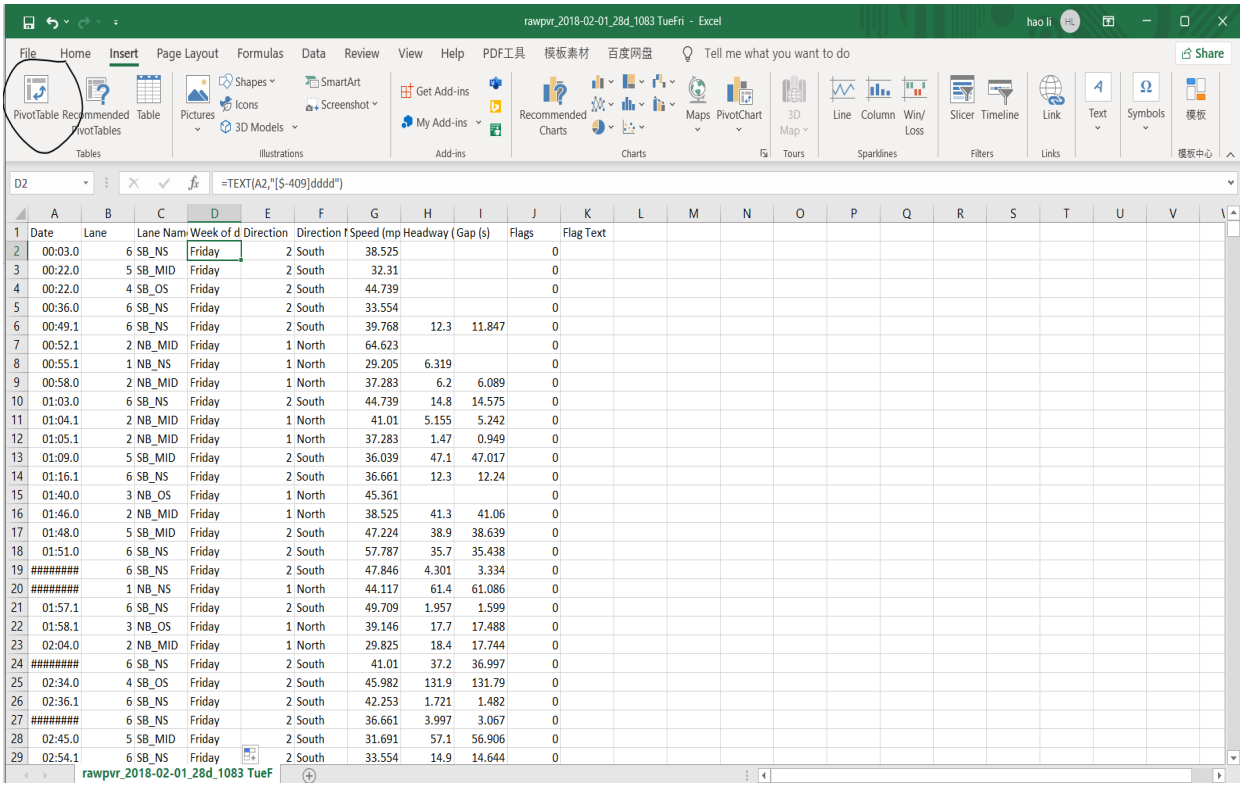



Figure 30. Insert PivotTable

Add “Filter”,”rows” and “values” information in the PivotTable field. Since we need to filter with time range(17:00-18:00), week of the day(Friday) and Lane name. We add these three to the “Filter” field and date to “Row” field. We need to get speed as value. Then add speed to the value field.

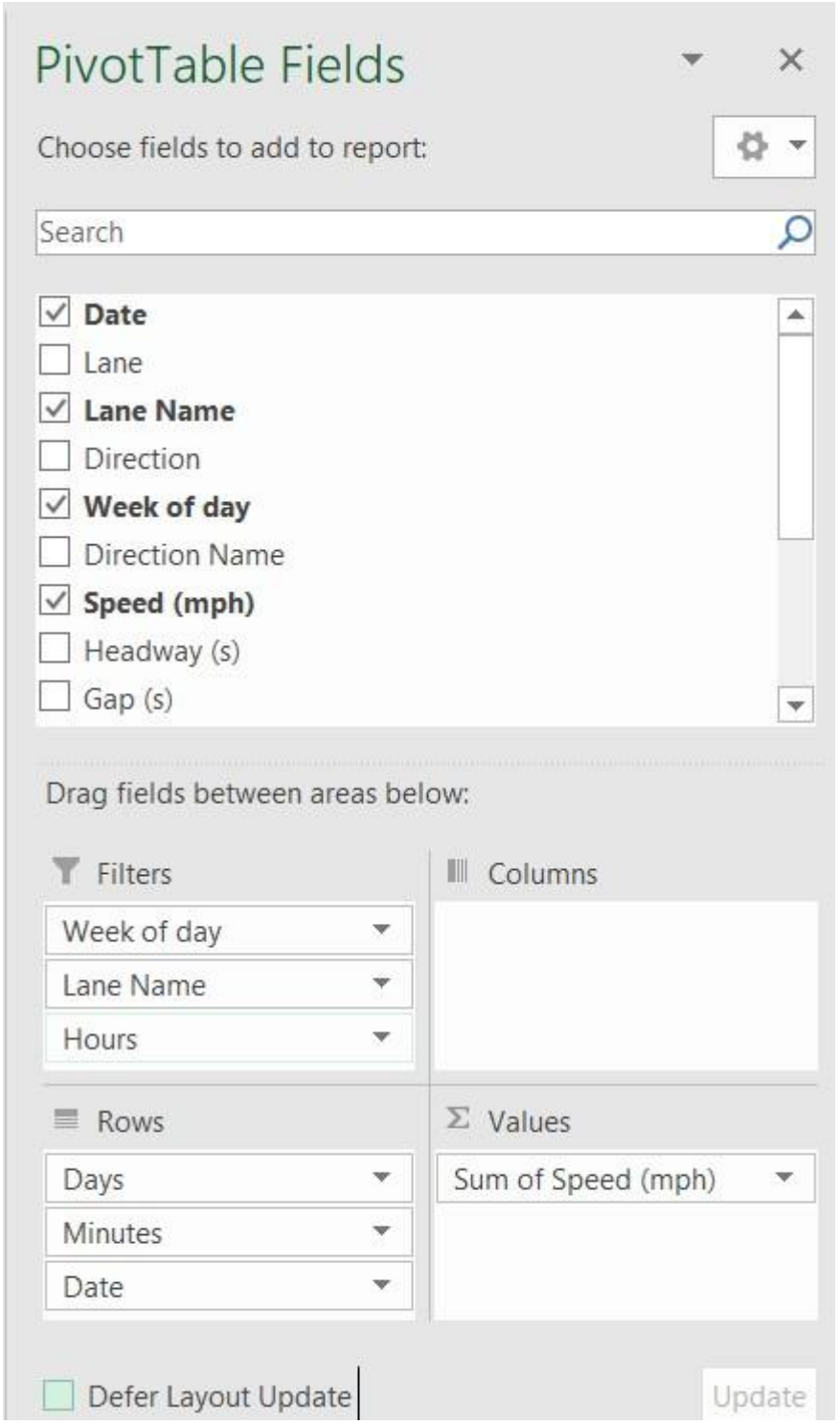
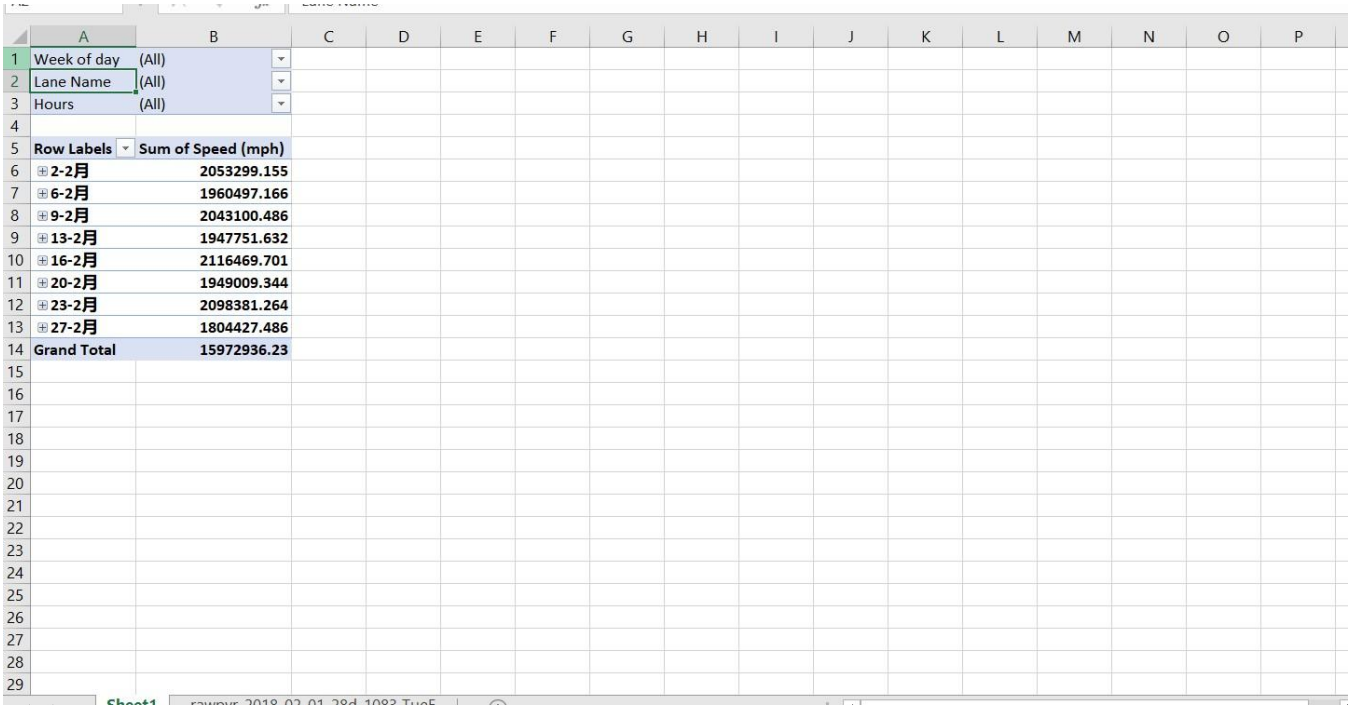



Figure 31. PivotTable field

Select item in filter.



| | A | B |
|---|---|---|
| 1 | Week of day | (All) |
| 2 | Lane Name | (All) |
| 3 | Hours | (All) |
| 4 | | |
| 5 | **Row Labels** | **Sum of Speed (mph)** |
| 6 | **2-2月** | **2053299.155** |
| 7 | **6-2月** | **1960497.166** |
| 8 | **9-2月** | **2043100.486** |
| 9 | **13-2月** | **1947751.632** |
| 10 | **16-2月** | **2116469.701** |
| 11 | **20-2月** | **1949009.344** |
| 12 | **23-2月** | **2098381.264** |
| 13 | **27-2月** | **1804427.486** |
| 14 | **Grand Total** | **15972936.23** |

Figure 32. PivotTable

Filter with Friday

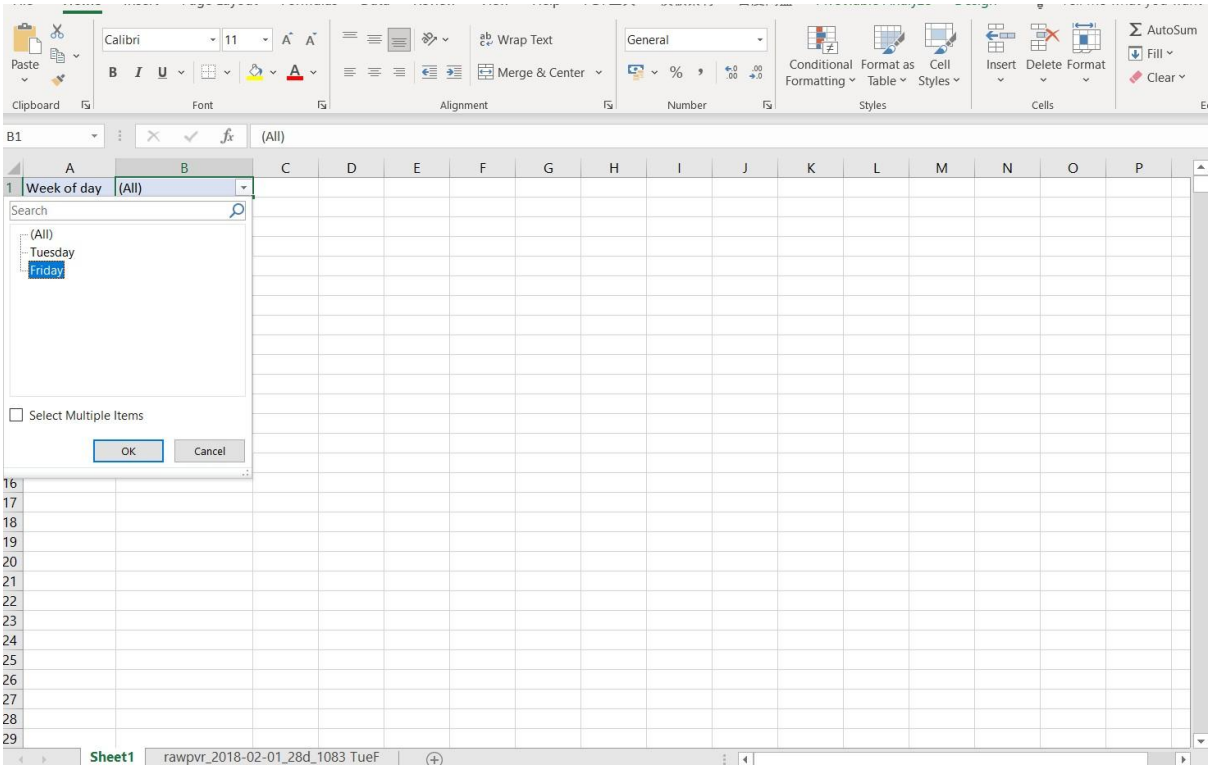



Figure 33. PivotTable week of day

# Filter with lane name(NB_MID as first example)

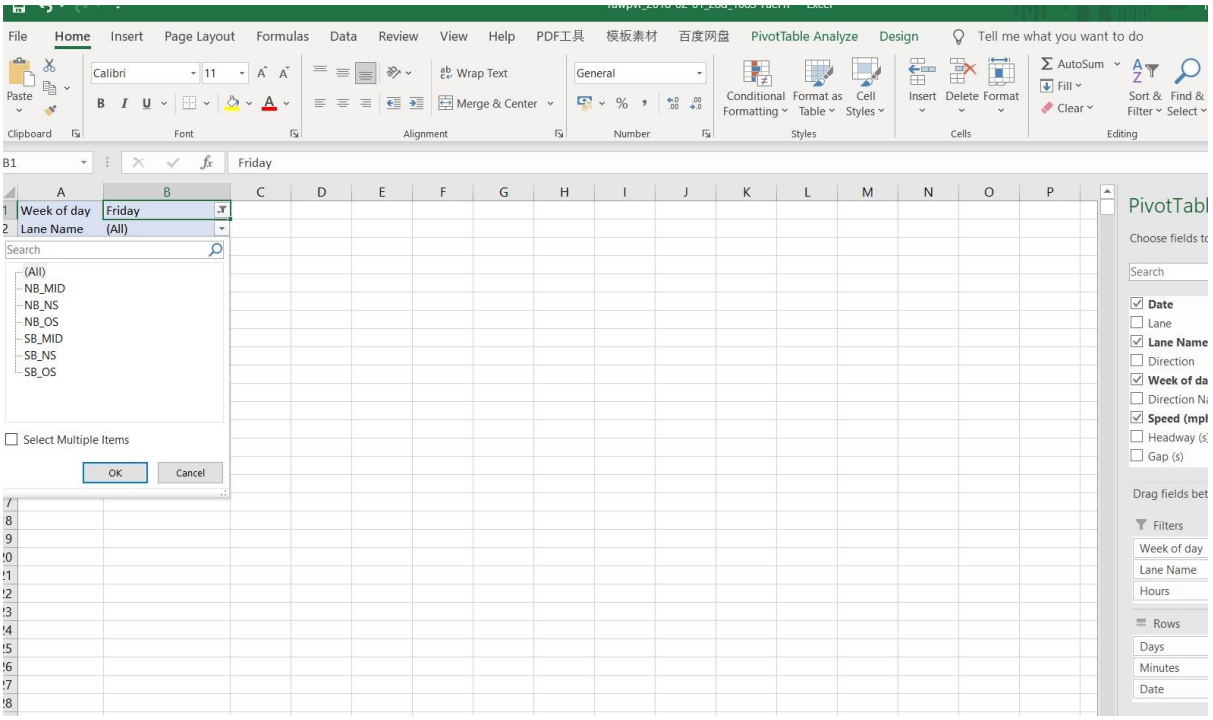



Figure 33. PivotTable lane name

Time range 17:00-18:00

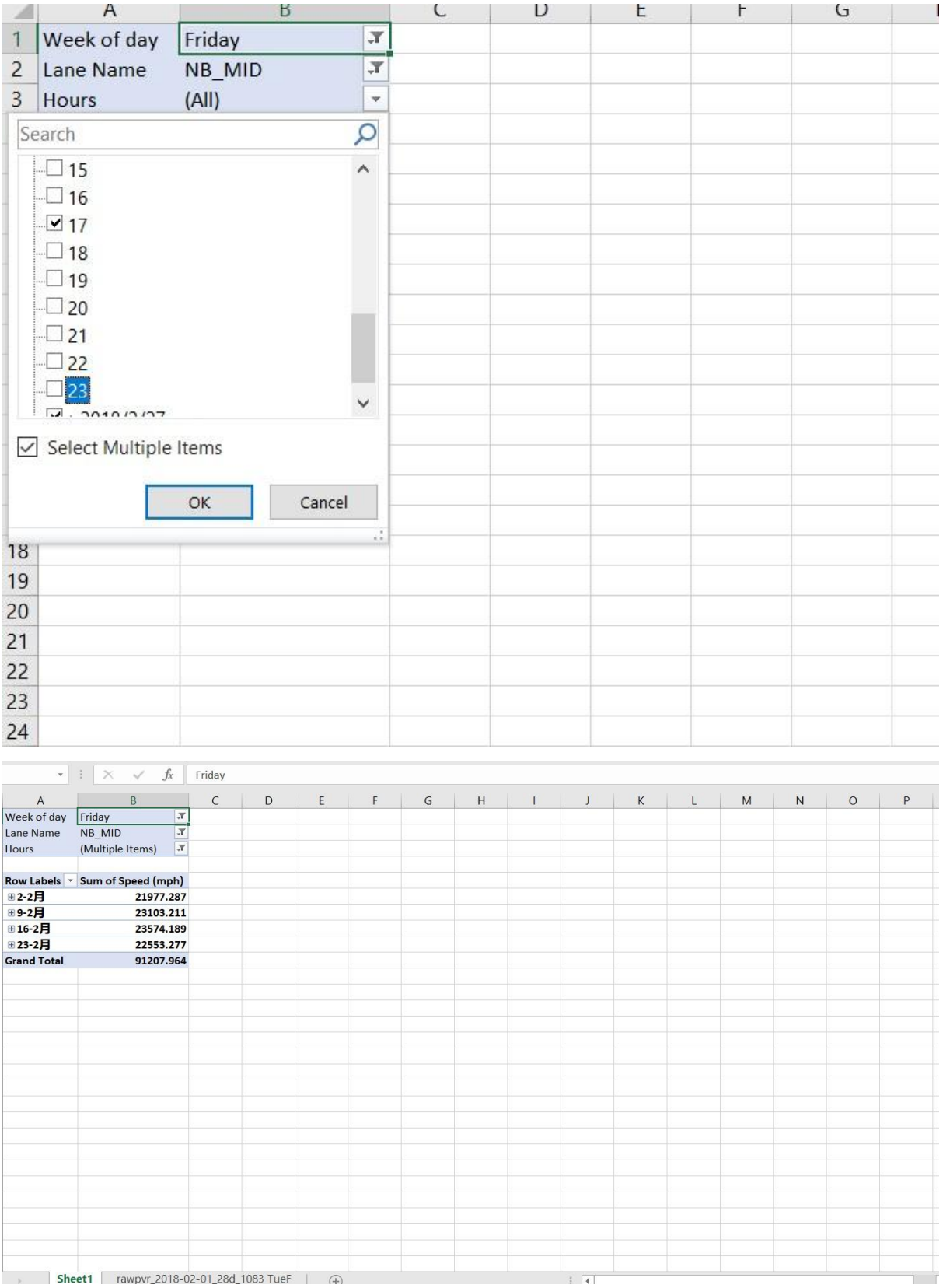



Figure 34. PivotTable time range

Go back to the value field setting part to set as average speed.

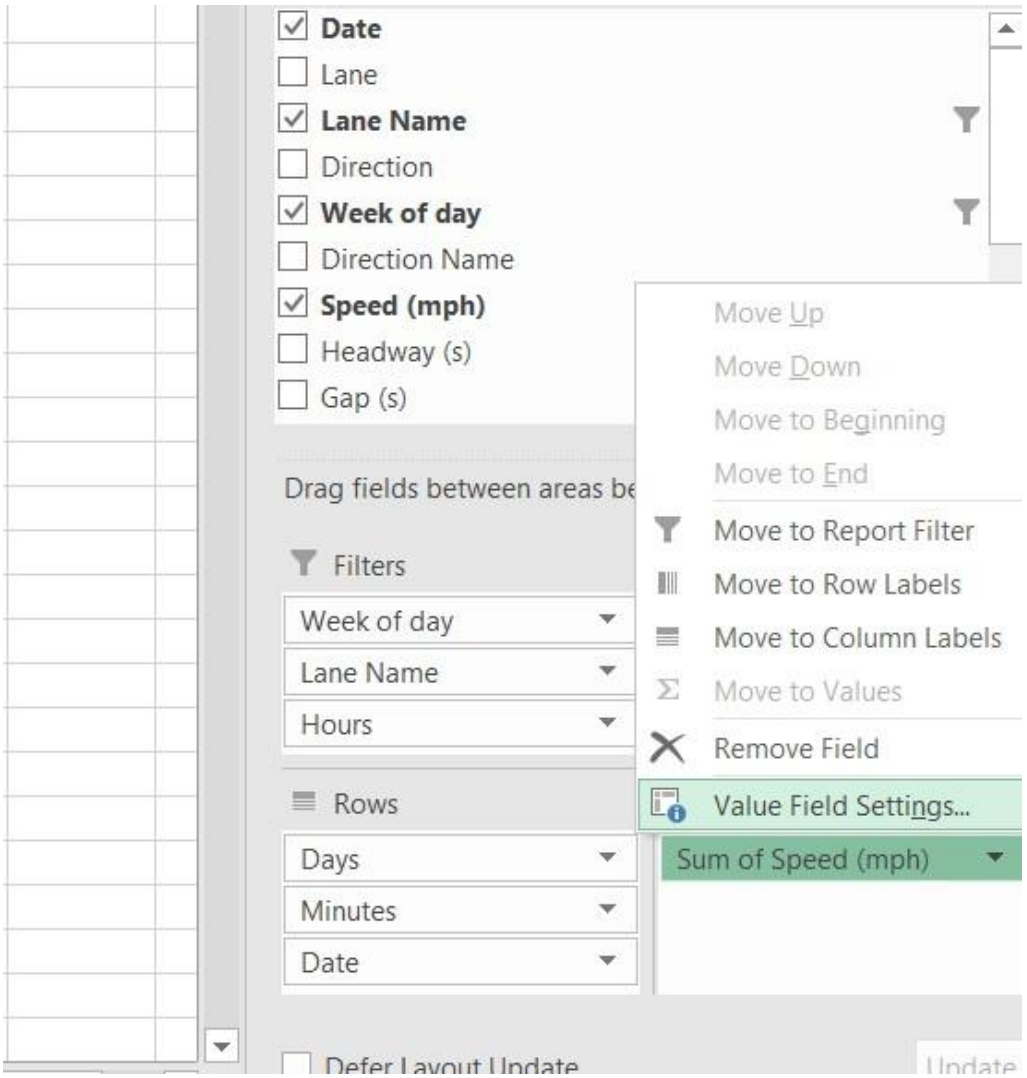



Figure 35. PivotTable value field setting

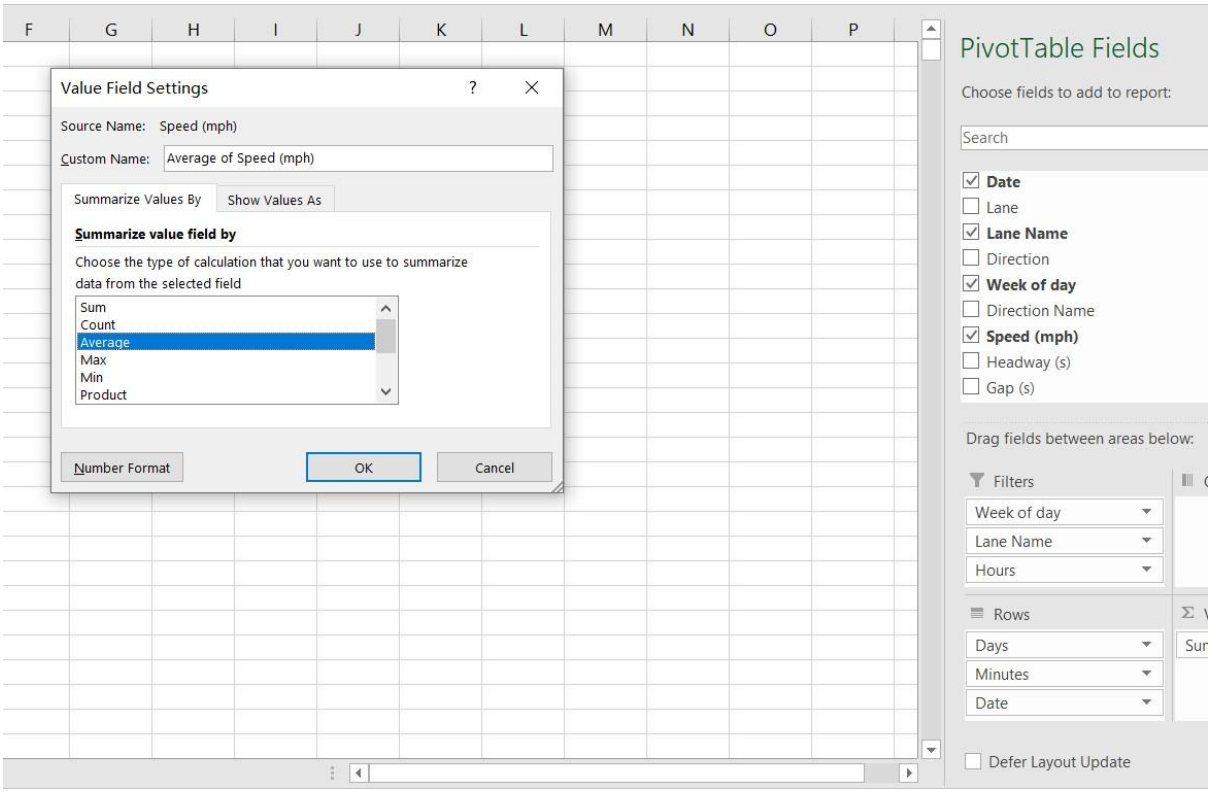
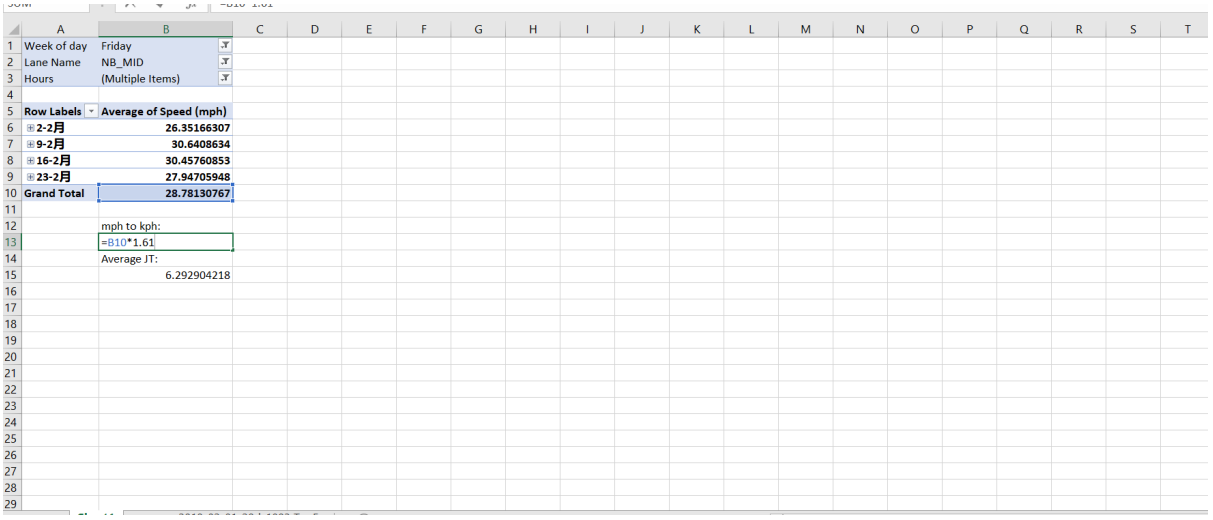



Figure 36. choose averages

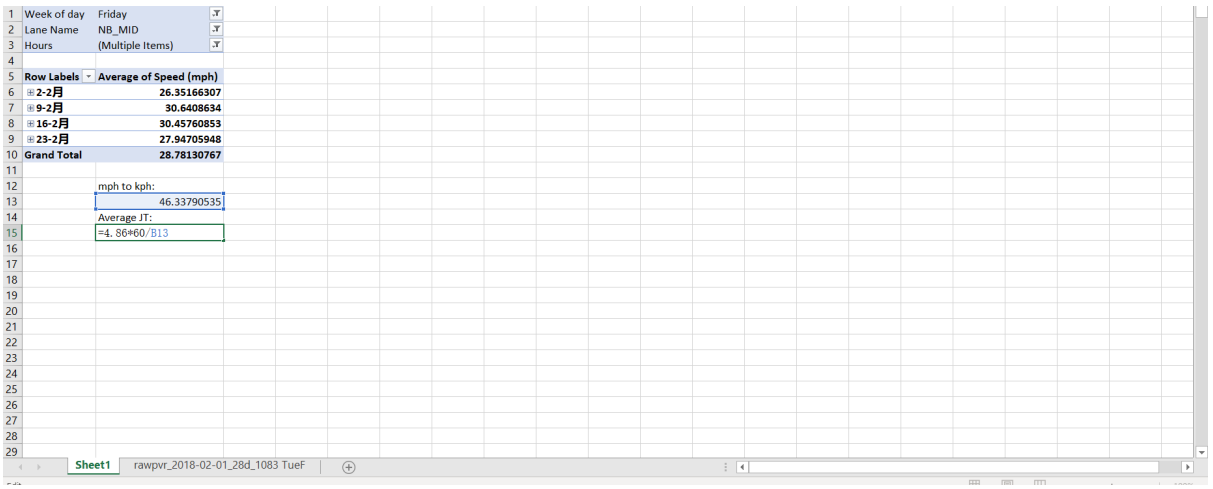
## Convert mph to kph(From task6 we could know 1 mph =1.6kph)



| | A | B |
|---|---|---|
| 1 | Week of day | Friday |
| 2 | Lane Name | NB_MID |
| 3 | Hours | (Multiple Items) |
| 4 | | |
| 5 | Row Labels | Average of Speed (mph) |
| 6 | 2-2月 | 26.35166307 |
| 7 | 9-2月 | 30.6408634 |
| 8 | 16-2月 | 30.45760853 |
| 9 | 23-2月 | 27.94705948 |
| 10 | Grand Total | 28.78130767 |
| 11 | | |
| 12 | | mph to kph: |
| 13 | | =B10*1.61 |
| 14 | | Average JT: |
| 15 | | 6.292904218 |

Figure 37. Convert mph to kph

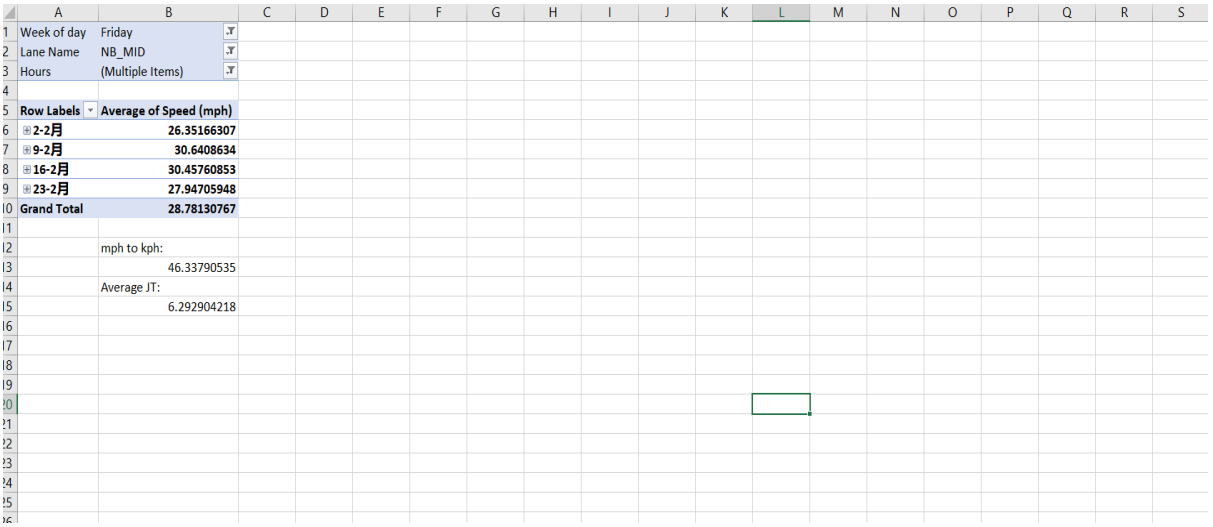
## Use this function to compute the average journey time on Friday.



| | | |
|---|---|---|
| 1 | Week of day | Friday |
| 2 | Lane Name | NB_MID |
| 3 | Hours | (Multiple Items) |
| 4 | | |
| 5 | **Row Labels** | **Average of Speed (mph)** |
| 6 | **2-2月** | **26.35166307** |
| 7 | **9-2月** | **30.6408634** |
| 8 | **16-2月** | **30.45760853** |
| 9 | **23-2月** | **27.94705948** |
| 10 | **Grand Total** | **28.78130767** |
| 11 | | |
| 12 | | mph to kph: |
| 13 | | 46.33790535 |
| 14 | | Average JT: |
| 15 | | =4. 86*60/B13 |
| 16 | | |
| 17 | | |
| 18 | | |
| 19 | | |
| 20 | | |
| 21 | | |
| 22 | | |
| 23 | | |
| 24 | | |
| 25 | | |
| 26 | | |
| 27 | | |
| 28 | | |
| 29 | | |

Sheet1 | rawpvr_2018-02-01_28d_1083 TueF

Figure 38. Average function



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Week of day | Friday | | | | | | | | | | | | | | | | | |
| 2 | Lane Name | NB_MID | | | | | | | | | | | | | | | | | |
| 3 | Hours | (Multiple Items) | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | |
| 5 | **Row Labels** | **Average of Speed (mph)** | | | | | | | | | | | | | | | | | |
| 6 | **2-2月** | **26.35166307** | | | | | | | | | | | | | | | | | |
| 7 | **9-2月** | **30.6408634** | | | | | | | | | | | | | | | | | |
| 8 | **16-2月** | **30.45760853** | | | | | | | | | | | | | | | | | |
| 9 | **23-2月** | **27.94705948** | | | | | | | | | | | | | | | | | |
| 10 | **Grand Total** | **28.78130767** | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | | |
| 12 | | mph to kph: | | | | | | | | | | | | | | | | | |
| 13 | | 46.33790535 | | | | | | | | | | | | | | | | | |
| 14 | | Average JT: | | | | | | | | | | | | | | | | | |
| 15 | | 6.292904218 | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | | |

Figure 39. Average output

Then filter other lane name to get the average speed

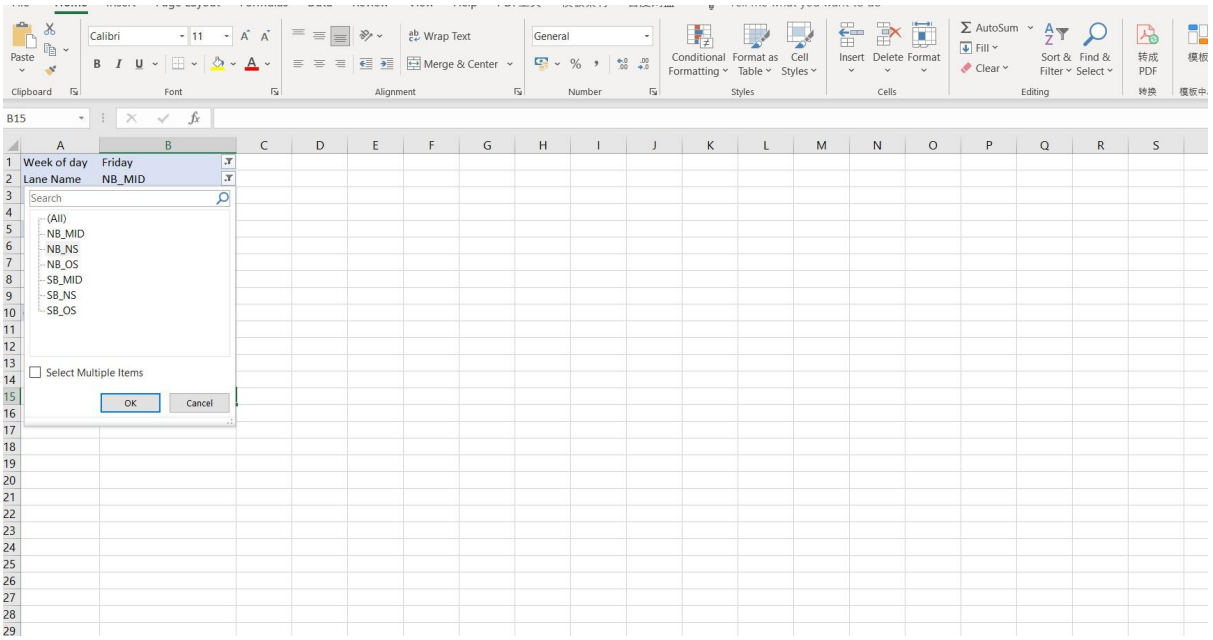



Figure 40. NBMID output

(NB_NS)

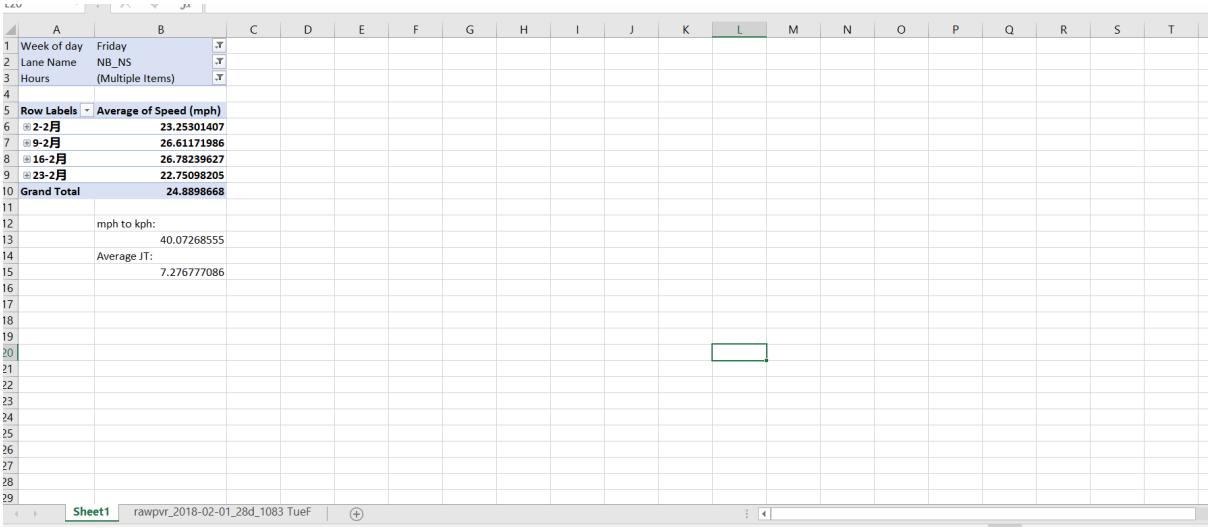
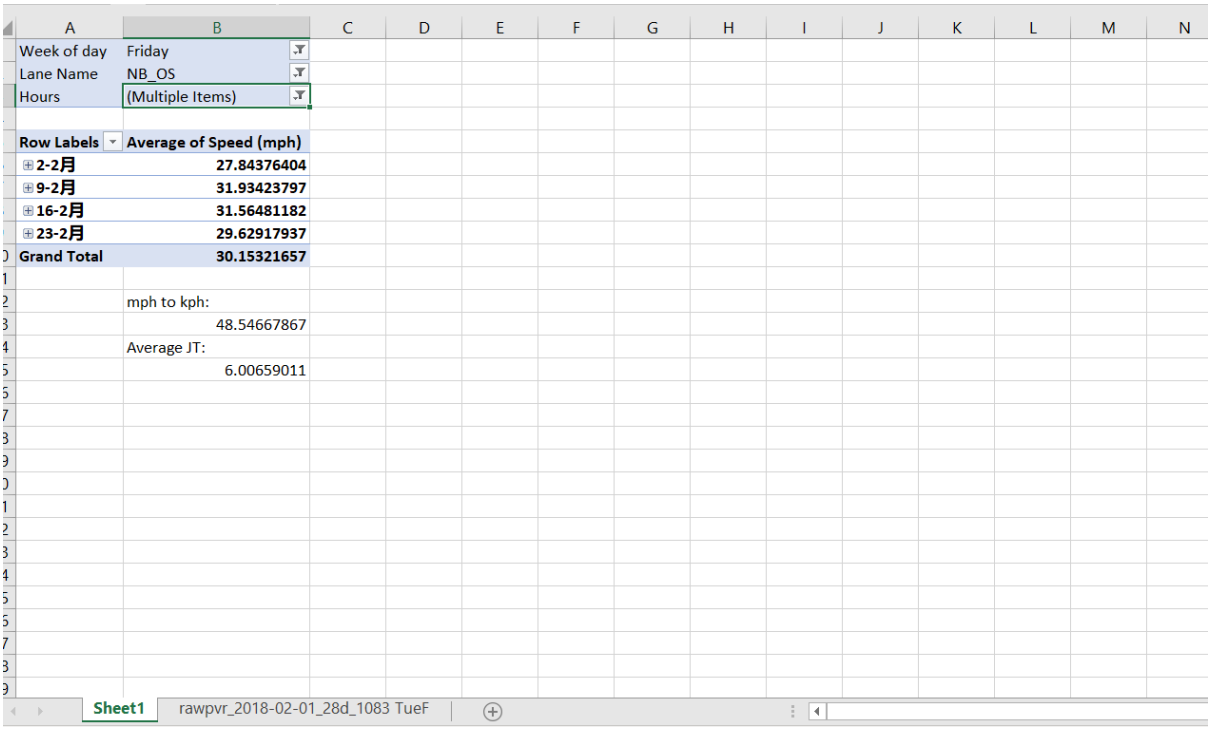



Figure 41. NBNS output

(NB_OS)



| | |
|---|---|
| Week of day | Friday |
| Lane Name | NB_OS |
| Hours | (Multiple Items) |

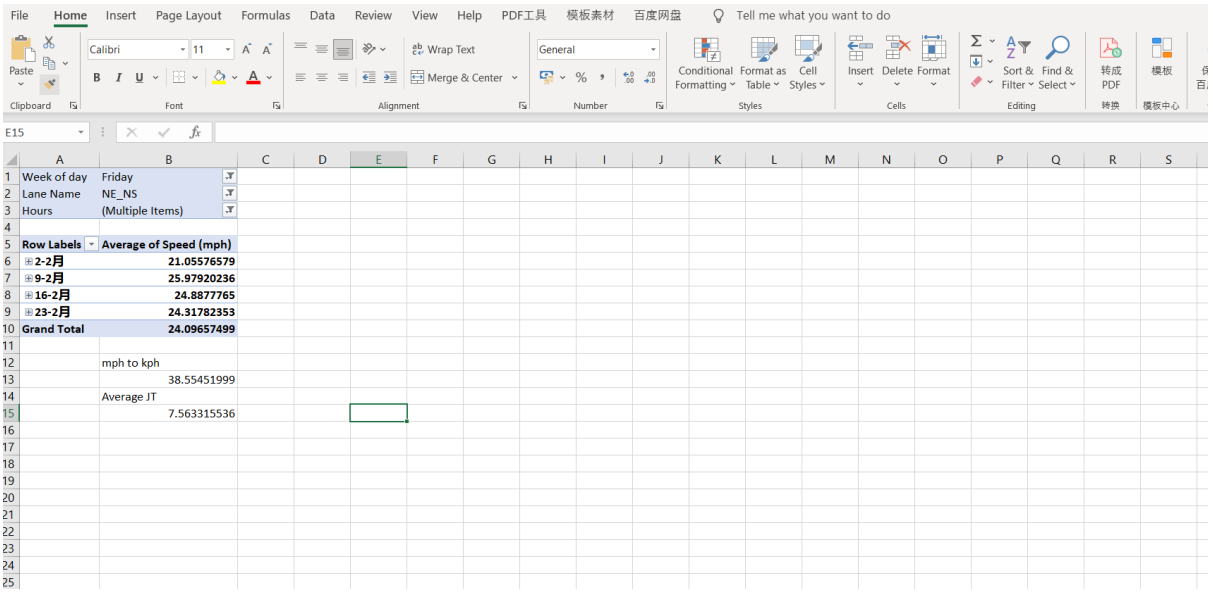
| Row Labels | Average of Speed (mph) |
|---|---|
| 2-2月 | 27.84376404 |
| 9-2月 | 31.93423797 |
| 16-2月 | 31.56481182 |
| 23-2月 | 29.62917937 |
| Grand Total | 30.15321657 |

mph to kph: 48.54667867

Average JT: 6.00659011

Figure 42. NBOS output

site 1415, NE_NS



| | |
|---|---|
| Week of day | Friday |
| Lane Name | NE_NS |
| Hours | (Multiple Items) |

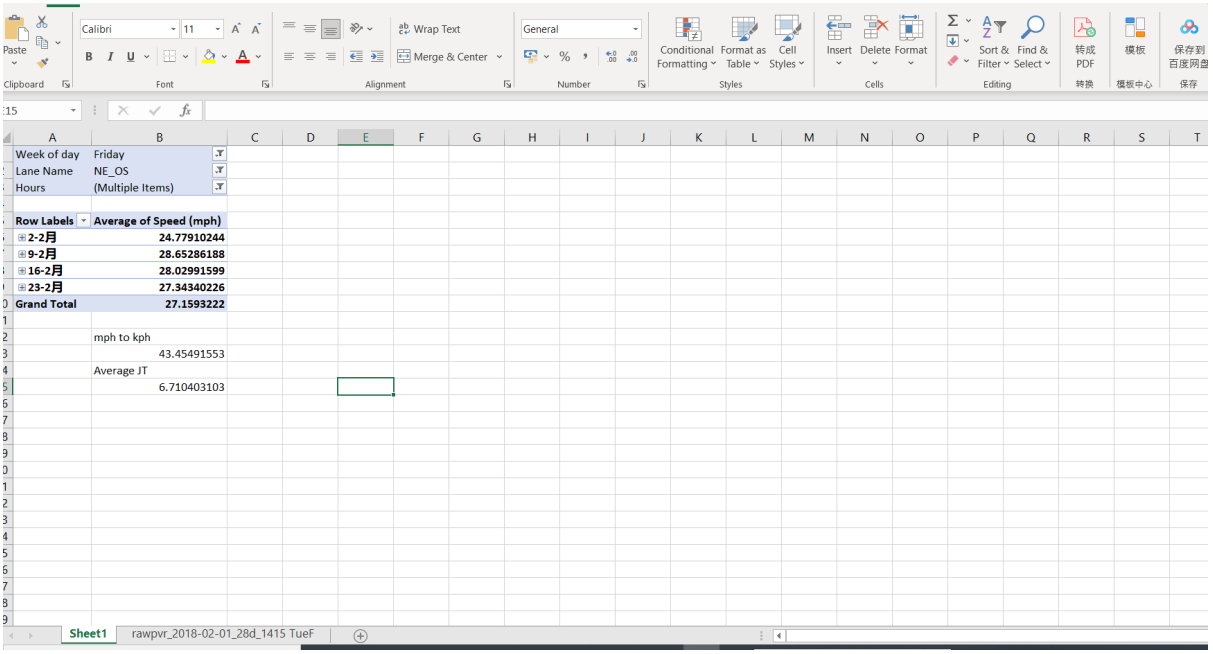
| Row Labels | Average of Speed (mph) |
|---|---|
| 2-2月 | 21.05576579 |
| 9-2月 | 25.97920236 |
| 16-2月 | 24.8877765 |
| 23-2月 | 24.31782353 |
| Grand Total | 24.09657499 |

mph to kph: 38.55451999

Average JT: 7.563315536

Figure 43. NENS output

Site 1415 NE_OS



| | |
|---|---|
| Week of day | Friday |
| Lane Name | NE_OS |
| Hours | (Multiple Items) |

| Row Labels | Average of Speed (mph) |
|---|---|
| 2-2月 | 24.77910244 |
| 9-2月 | 28.65286188 |
| 16-2月 | 28.02991599 |
| 23-2月 | 27.34340226 |
| Grand Total | 27.1593222 |

| | |
|---|---|
| mph to kph | |
| | 43.45491553 |
| Average JT | |
| | 6.710403103 |

Figure 44. NEOS output

# Output:

Therefore, these value would be {6.29(site1083_NBMID),7.28(site1083_NBNS),6(site1083_NBOS),7.56(site1415_NENS),6.71(site1415_NE_OS)
{6.29,7.28,6,7.56,6.71}
The average is 6.768 which is close to 6.75 in task 6.

# Discussion:

For this task, Both technologies have their own advantage.
For python, firstly, the task asked the user to process data in two different CSV files. To process data from two separate files are hard for EXCEL. However, python only requires users to make one more variable for reading data to solve this problem. Secondly, the time range filter is not very easy to use in excel.

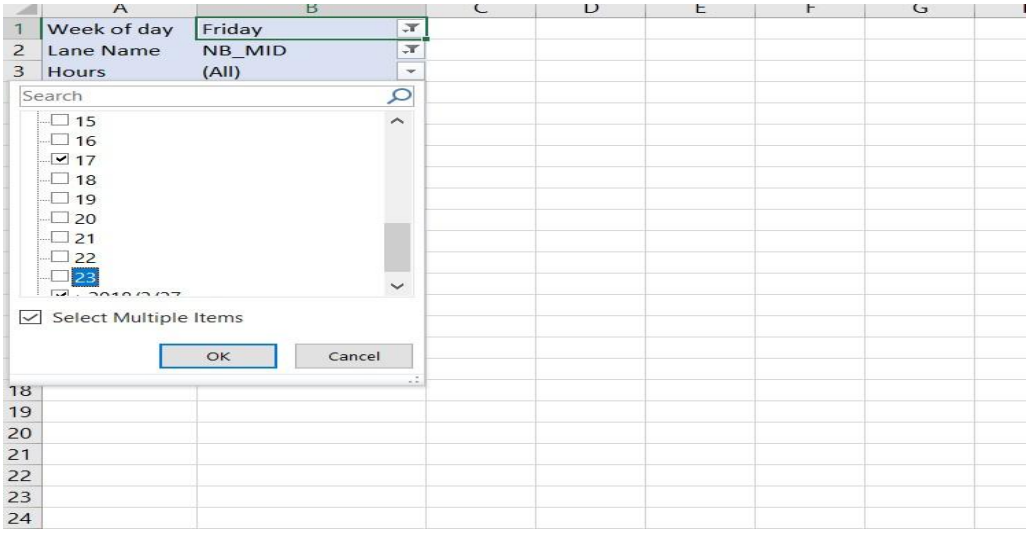



Figure 45. example1

As the image is shown before, the requirement asks the user to collect data in 17:00-18:00 only. However, users have to select multiple items filter and cancel other time ranges. In python, the problem have been solved by very easy code:

```
mid1083 = Date_with_hour.between(pd.Timedelta('17:00:00'), pd.Timedelta('18:00:00'),inclusive="left")
```

Figure 46. example2

Excel also has some advantages for operation. For instance, in this lane name filter part is shown below.

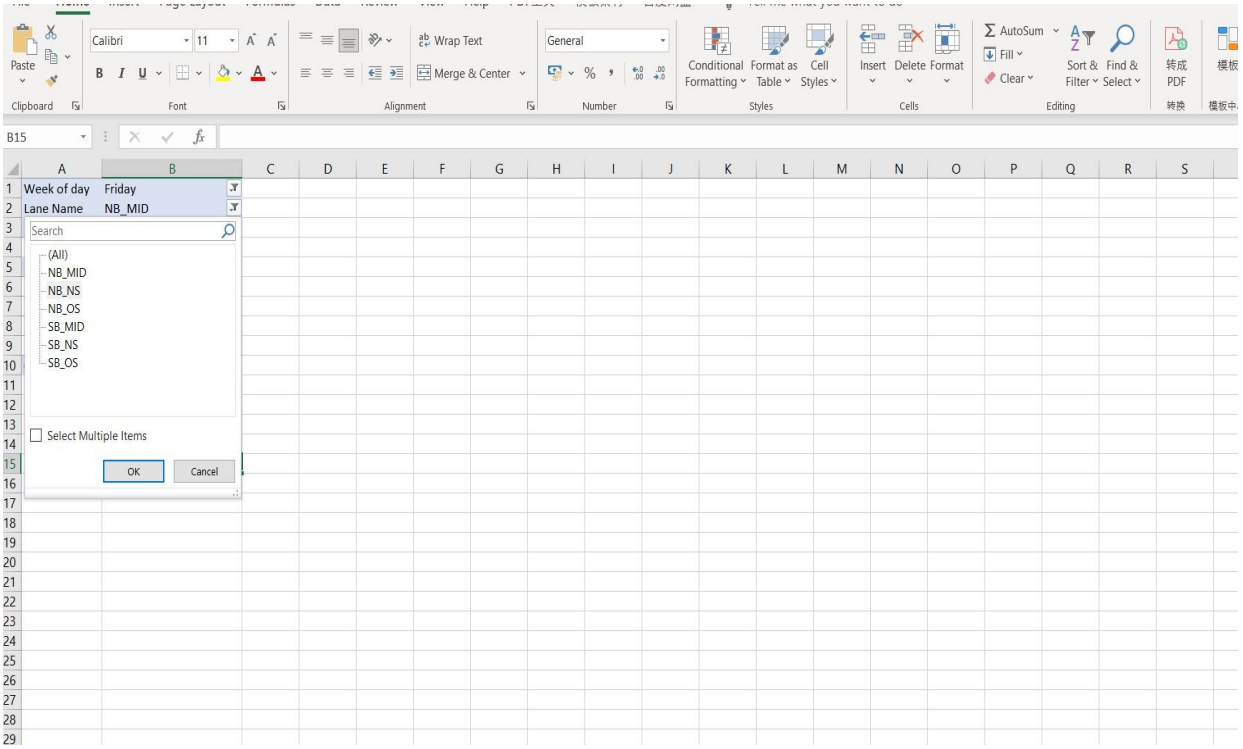



Figure 47. example3

Changing the lane name is easy to use in Excel. The user could select the “NB_MID”,”NB_NS”,”NB_OS” to change this condition.
However, in python code here:

```
69  site1415_ne = list3.loc[(list3['week of day'] == 4)&(list3['Lane Name']== "NE_OS")]
70
71  Date_information_NE = pd.to_datetime(site1415_ne.Date)
72  Date_with_day_NE = Date_information_NE.dt.floor('D')
73  Date_with_hour_NE = Date_information_NE - Date_with_day_NE
74  ne1415 = Date_with_hour_NE.between(pd.Timedelta('17:00:00'), pd.Timedelta('18:00:00'),inclusive="left")
75  ne1415_17_18 = site1415_ne.loc[ne1415]
76  ne1415_speed = ne1415_17_18['Speed (mph)']
77  average_ne_speed = ne1415_speed.mean(axis=0)
78  ne_kph = average_ne_speed*1.61
79  ne_result = (4.86*60)/ne_kph
80  print(ne_result)
81  list_result.append(ne_result)
82
83  site1415_nens = list3.loc[(list3['week of day'] == 4)&(list3['Lane Name']== "NE_NS")]
84
85  Date_information_NENS = pd.to_datetime(site1415_nens.Date)
86  Date_with_day_NENS = Date_information_NENS.dt.floor('D')
87  Date_with_hour_NENS = Date_information_NENS - Date_with_day_NENS
88  nens1415 = Date_with_hour_NENS.between(pd.Timedelta('17:00:00'), pd.Timedelta('18:00:00'),inclusive="left")
89  nens1415_17_18 = site1415_nens.loc[nens1415]
90  nens1415_speed = nens1415_17_18['Speed (mph)']
91  average_nens_speed = nens1415_speed.mean(axis=0)
92  nens_kph = average_nens_speed*1.61
93  nens_result = (4.86*60)/nens_kph
94  print(nens_result)
95  list_result.append(nens_result)
```

Figure 48. example4

Due to the limitation of the current python code implementation, if the user wants to change the lane name condition. They have to select lane name again in line 83, repeat much of a very similar code from 85 to 93. It makes the variable name is very important in this case. If they put wrong or repeat the variable name. Then it is very likely to lead to programming errors and data errors.

Reference:

[1] Metric Conversions, [Online],Available at: https://www.metric-conversions.org/speed/miles-per-hour-to-kilometers-per-hour.htm [Accessed 21/10/2021]
[2] Oracle Enterprise Data Quality Online Help, [Online], Available at: https://docs.oracle.com/en/middleware/fusion-middleware/enterprise-data-quality/12.2.1.4/edqoh/record-completeness-profiler.html [Accessed 21/10/2021]