

Task5

- **Task5.1**

To get the Column Completeness, the data need to be filtered out according to the conditions. Because the conditions of 'Headway(s)' and 'Gap(s)' are the same, they can be filtered together.

```
# use a dictionary to store the Column Completeness.
Column_Completeness = {}

# get all data on Tuesday. 'Flags' refers to the day of the week, which has
# been gotten in task 1.
condition = dataset.loc[dataset['Flags'] == 2]

# Extract the time from 7:00 to 19:00. 'Hours' refers to the hours of a day.
condition = condition.loc[(condition['Hours'] >= 7) & (condition['Hours'] <
19)]

# get the quality assessment of the level of completeness of the 'Gaps(s)'.
# condition['Gap (s)'].count() is used for calculating the non-empty value
of 'Gap (s)'
# condition['Gap (s)'].size is used for getting number of the cells.
Column_Completeness['Gap (s)'] = condition['Gap (s)'].count() * 100 /
condition['Gap (s)'].size

# get the quality assessment of the level of completeness of the
'Headways(s)'. 'Hours' refers to the hours of a day, which I defined in task
2.
# condition['Headway (s)'].count() is used for calculating the non-empty
value of 'Headway (s)'
# condition['Headway (s)'].size is used for getting number of the cells.
Column_Completeness['Headway (s)'] = condition['Headway (s)'].count() * 100
/ condition['Headway (s)'].size

# show the result
Column_Completeness
```

Then the **final result** is as following:

```
Out[38]: {'Gap (s)': 98.03654443753885, 'Headway (s)': 98.96532007458049}
```

interpretation:

Completeness is one of the characteristics of data quality. In this task, it can be seen that in the 'Gap (s)' column, 98.04% of the data is non-empty. Besides, in the 'Headway (s)' column, 98.97% of the data is non-empty. I think the level of completeness for 'Gap (s)' and 'Headway (s)' is very high, but not 100, that means there are still some data is empty.

Therefore, it would be better for us to do data cleaning to improve data quality.

- **Task5.2**

- **The first step:** Get median.

```
# get the lane named NB_MID
NB_MID = dataset.loc[dataset['Lane Name'] == 'NB_MID']

# get Tuesday
NB_MID = NB_MID.loc[NB_MID['Flags'] == 2]

# get the median between 7:00 - 19:00
NB_MID_median = NB_MID.loc[(NB_MID['Hours'] >= 7) & (NB_MID['Hours'] < 19)].groupby(['Hours']).median()
NB_MID_median
```

	Lane	Direction	Speed (mph)	Headway (s)	Gap (s)	Flags	Day_of_the_month
Hours							
7	2	1	19.2620	2.7220	1.8340	2	13
8	2	1	15.2235	3.1910	2.1020	2	20
9	2	1	28.5840	2.7200	2.0800	2	13
10	2	1	31.6910	3.0000	2.5835	2	13
11	2	1	32.3100	3.2100	2.7785	2	13
12	2	1	32.3100	3.1600	2.7950	2	13
13	2	1	32.3100	3.1020	2.7560	2	13
14	2	1	32.3100	3.1330	2.8050	2	13
15	2	1	31.6910	2.9200	2.5770	2	20
16	2	1	27.9620	2.8530	2.3225	2	20
17	2	1	24.8550	2.9065	2.1870	2	13
18	2	1	28.5840	2.7900	2.2280	2	13

From the picture above, 'Gap (s)' and 'Headways' can be seen.

The medians of 'Gap (s)' and 'Headway (s)' are as following:

Hours	Gap(s)	Headway(s)
7	1.8340	2.7220
8	2.1020	3.1910
9	2.0800	2.7200
10	2.5835	3.0000
11	2.7785	3.2100
12	2.7950	3.1600
13	2.7560	3.1020
14	2.8050	3.1330
15	2.5770	2.9200
16	2.3225	2.8530
17	2.1870	2.9065
18	2.2280	2.7900

- **The second step:**

Write a function to fill the median of 'Gap (s)' and 'Headway (s)'.

In the description of this task, it is said that we can get the median by sorting, but there is no need for sorting steps, because I use pandas.

Function to fill the median of 'Gap (s)' :

```
# function to fill median of 'Gap (s),' x represents each row in the
dataset.
def fill_median_of_gaps(x):

    # If the name of the road is not 'NB_MID' or 'Gap(s)' is not a null
    value or the time is not in 7:00-19:00, return to the original value.
    if x['Lane Name'] != 'NB_MID' or not pd.isna(x['Gap (s)']) or
x['Hours'] >= 19 or x['Hours'] < 7:
        return x

    # else use the median to fill the missing data.
    # use NB_MID_median.loc[int(x['Hours'])] to get the Hours of x, and
    find the median of the corresponding 'Gap(s)'.
    x['Gap (s)'] = NB_MID_median.loc[int(x['Hours'])]['Gap (s)']
    return x

# assign to dataset. Now the empty columns of 'Gap (s)' is filled with
median.
dataset = dataset.apply(fill_median_of_gaps, axis=1)
```

In the same way, function to fill the median of 'Headway (s)' are as followed:

```
# function to fill median of 'Headway (s),' x represents each row in the
dataset.
def fill_median_of_headways(x):
```

```

        # If the name of the road is not 'NB_MID' or 'Headway (s)' is not a
        null value or the time is not in 7:00-19:00, return to the original
        value.
        if x['Lane Name'] != 'NB_MID' or not pd.isna(x['Headway (s)']) or
        x['Hours'] >= 19 or x['Hours'] < 7:
            return x

        # else use the median to fill the missing data.
        # use NB_MID_median.loc[int(x['Hours'])] to get the Hours of x, and
        find the median of the corresponding # else use median to fill the
        missing data.
        # use NB_MID_median.loc[int(x['Hours'])] to get the Hours of x, and
        find the median of the corresponding 'Gap(s)'.
        x['Headway (s)'] = NB_MID_median.loc[int(x['Hours'])]['Headway (s)']
        return x

# assign to dataset. Now the empty columns of 'Gap (s)' is filled with
median.
dataset = dataset.apply(fill_median_of_headways, axis=1)

```

In order to judge whether the code is correct, there are two snapshots below, one stands for some data before executing these two functions, and another stands for the same data after executing these two functions. By doing this, a comparison can be made.

Before executing:

I selected all the data at 7 o'clock, and the lane name called NB_MID and Gap(s) and Headway(s) are missing.

```

dataset.loc[(dataset['Hours'] == 7) & (dataset['Lane Name'] == 'NB_MID')
& (pd.isna(dataset['Gap (s)'])) & (pd.isna(dataset['Headway (s)']))]

```

We can see the part of these data

	Date	Lane	Lane Name	Direction	Direction Name	Speed (mph)	Headway (s)	Gap (s)	Flags	Flag Text	Hours	Day_of_the_month
3947	2018-02-02 07:00:27.090000	2	NB_MID	1	North	27.340	NaN	NaN	5	Friday	7	2
4249	2018-02-02 07:05:40.050000	2	NB_MID	1	North	42.253	NaN	NaN	5	Friday	7	2
4561	2018-02-02 07:10:21.070000	2	NB_MID	1	North	39.768	NaN	NaN	5	Friday	7	2

We could also choose some data at 9 o'clock, and the lane name called NB_MID and Gap(s) and Headway(s) are missing.

	Date	Lane	Lane Name	Direction	Direction Name	Speed (mph)	Headway (s)	Gap (s)	Flags	Flag Text	Hours	Day_of_the_month
14525	2018-02-02 09:10:39.070000	2	NB_MID	1	North	33.554	NaN	NaN	5	Friday	9	2
14845	2018-02-02 09:15:20.080000	2	NB_MID	1	North	39.768	NaN	NaN	5	Friday	9	2
15236	2018-02-02 09:21:01.060000	2	NB_MID	1	North	42.253	NaN	NaN	5	Friday	9	2

After executing:

Compared to the average that we got, we can say that we got the correct answer.

Hours = 7:

	Date	Lane	Lane Name	Direction	Direction Name	Speed (mph)	Headway (s)	Gap (s)	Flags	Flag Text	Hours	Day_of_the_month
3947	2018-02-02 07:00:27.090000	2	NB_MID	1	North	27.340	2.722	1.834	5	Friday	7	2
4249	2018-02-02 07:05:40.050000	2	NB_MID	1	North	42.253	2.722	1.834	5	Friday	7	2
4561	2018-02-02 07:10:21.070000	2	NB_MID	1	North	39.768	2.722	1.834	5	Friday	7	2

Hours = 9:

	Date	Lane	Lane Name	Direction	Direction Name	Speed (mph)	Headway (s)	Gap (s)	Flags	Flag Text	Hours	Day_of_the_month
14525	2018-02-02 09:10:39.070000	2	NB_MID	1	North	33.554	2.72	2.08	5	Friday	9	2
14845	2018-02-02 09:15:20.080000	2	NB_MID	1	North	39.768	2.72	2.08	5	Friday	9	2
15236	2018-02-02 09:21:01.060000	2	NB_MID	1	North	42.253	2.72	2.08	5	Friday	9	2

- **interpretation:**

Filling in empty values with median is a common data processing method, we can also use mean or mode to fill it. What we are doing now is called data cleaning, which can improve data quality.

For the final result, if we do not want 'Hours' and 'Day_of_the_month.' We could use del to delete it, and this helps to improve data quality because these two columns are actually redundant data. For the convenience of processing data later, I have not deleted it temporarily, but it should be deleted when the data is stored.

Task6

- **The first step:** Get the average speed at 17:00 on Fridays that the direction is north.

In file 1083, the code is as following:

```
# find the data on Friday
condition_for_avg_speed = dataset.loc[dataset['Flags'] == 5]

# find the data that between 17:00 - 18:00
condition_for_avg_speed = condition_for_avg_speed.loc[dataset['Hours'] == 17]

# find the data that direction is north
condition_for_avg_speed = condition_for_avg_speed.loc[dataset['Direction'] == 1]

# get the average speed of 1083
avg_1083 = condition_for_avg_speed.groupby(['Lane']).mean()
```

The answer is as followed:

	Direction	Speed (mph)	Headway (s)	Gap (s)	Flags	Hours	Day_of_the_month
Lane							
1	1.0	24.889867	4.464115	3.520640	5.0	17.0	12.128864
2	1.0	28.781308	4.446057	3.974463	5.0	17.0	12.432629
3	1.0	30.153217	4.320693	3.821315	5.0	17.0	12.468373

It can be seen that the averages of each lane, and it can be used by using `avg_1083['Speed (mph)']`

In the file 1415, the data should be put in pandas first.

```
data_1415 = pd.read_csv("rawpvr_2018-02-01_28d_1415 TueFri.csv", index_col =
False)
dataset.head()
```

	Date	Lane	Lane Name	Direction	Direction Name	Speed (mph)	Headway (s)	Gap (s)	Flags	Flag Text	Hours	Day_of_the_month
0	2018-02-02 00:00:03.050000	6	SB_NS	2	South	38.525	NaN	NaN	5	Friday	0	2
1	2018-02-02 00:00:22.010000	5	SB_MID	2	South	32.310	NaN	NaN	5	Friday	0	2
2	2018-02-02 00:00:22.020000	4	SB_OS	2	South	44.739	NaN	NaN	5	Friday	0	2
3	2018-02-02 00:00:36.040000	6	SB_NS	2	South	33.554	NaN	NaN	5	Friday	0	2
4	2018-02-02 00:00:49.070000	6	SB_NS	2	South	39.768	12.3	11.847	5	Friday	0	2

Then, get the average number of speed, it can be seen that Direction = 2 stands for the Direction is north.

Besides, the label 'Flags' and the label 'Hours'(the hours of a day) should be gotten first.

```
# create a column to describe hour
data_1415['Hours'] = data_1415['Date'].apply(lambda s :
datetime.strptime(s[:19], "%Y-%m-%d %H:%M:%S").hour)

# get the Flags
data_1415['Flags'] = data_1415['Date'].apply(lambda s :
datetime.strptime(s[:19], "%Y-%m-%d %H:%M:%S").weekday()+1)
```

Then, select qualified data.

```
# find the data on Friday
condition_for_avg_speed = data_1415.loc[dataset['Flags'] == 5]

# find the data that between 17:00 - 18:00
condition_for_avg_speed = condition_for_avg_speed.loc[dataset['Hours'] ==
17]

# find the data that direction is north
condition_for_avg_speed = condition_for_avg_speed.loc[dataset['Direction']
== 1]

# get the average speed of 1083
avg_1415 = condition_for_avg_speed.groupby(['Lane']).mean()
```

We get the result.

	Direction	Speed (mph)	Headway (s)	Gap (s)	Flags	Flag Text	Hours
Lane							
1	1	24.096575	5.896678	4.597008	5	NaN	17
2	1	27.159322	6.336292	5.282062	5	NaN	17

Finally, calculate the average of the five lanes.

```
# /5 is because we got 5 speed.
avg_speed = (avg_1083['Speed (mph)'].sum() + avg_1415['Speed (mph)'].sum())
/ 5

# convert to mph -> km/h
avg_speed *= 1.6
```

The result is

```
Out[62]: 43.22569223365439
```

- **The second step:** Calculate the time.

```
# distance is given in task
distance = 4.86

time = distance / avg_speed

# get the result in minutes
time *= 60
```

So, the answer is

```
In [64]: time
```

```
Out[64]: 6.74598797455389
```

Convert to two decimal places is 6.75 (min.)

- **interpretation:**

In this task, we use average to describe the journey time for the north lanes between site 1083 and site 1415 between 17:00 and 18:00.

The average can describe the general level of the object. Through the average value, we can see where the data is concentrated. It can be used to get general information.

Task7

- **part1:**

An analogy to Task 5, I suggest 2 formula:

The **first** one is:

```
Rows_Completeness = (number_of_non-empty_cells_for_each_row x 100) /
number_of_cells
```

The **second** one is:

```
Another_Rows_Completeness = (number_of_non-empty_rows x 100) / number_of_row
```

I personally understand row completeness from two different perspectives. The first one is focusing on each specific row, measure the completeness of each specific row and express it by an average value, and the second is to treat each row as a unit to get the row completeness from the whole.

Since I have created two labels before, I need to delete them ('Hours', 'Day_of_the_month').

```
# del the label 'Hours' and 'Day_of_the_month'
tmp = condition.drop(['Hours', 'Day_of_the_month'], axis=1)
```

```
Out[59]:
```

	Date	Lane	Lane Name	Direction	Direction Name	Speed (mph)	Headway (s)	Gap (s)	Flags	Flag Text
0	2018-02-02 00:00:03.050000	6	SB_NS	2	South	38.525	NaN	NaN	5	Friday
1	2018-02-02 00:00:22.010000	5	SB_MID	2	South	32.310	NaN	NaN	5	Friday
2	2018-02-02 00:00:22.020000	4	SB_OS	2	South	44.739	NaN	NaN	5	Friday
3	2018-02-02 00:00:36.040000	6	SB_NS	2	South	33.554	NaN	NaN	5	Friday
4	2018-02-02 00:00:49.070000	6	SB_NS	2	South	39.768	12.300	11.847	5	Friday

It should be noticed that we should use the data before task 6 because task 6 has changed the data we deal with. Besides, I have created the variable 'condition' before, which is as followed.

```
# get all Tuesday. 'Flags' refers to the day of the week, which has been gotten in task 1.
condition = dataset.loc[dataset['Flags'] == 2]

# get the time between 7:00 - 19:00. 'Hours' refers to the hours of a day.
condition = condition.loc[(condition['Hours'] >= 7) & (condition['Hours'] < 19)]
```

According to the task description, we will only focus on 'Speed (mph)' and 'Headways (s)'. Therefore, it does not matter whether the 'Flags' and 'Flag Text' are empty (the initial data is empty but during the tasks, it changed, so it became non-empty), and there is no need to process them separately. Besides, if the 'Flags' and 'Flag Text' are not empty, the only two empty labels are 'Speed (mph)' and 'Headways (s)', that is what we needed.

We could use this condition to get completeness.

The first one is

```
# row.isna() means the cells that are missing (In this case, only highway
and speed could be missing).
# So len(row) - row.isna().sum() stands for the non-empty cells.
# len(row) can get all the cells in one row.
Rows_Completeness = tmp.apply(lambda row: (len(row) - row.isna().sum()) *
100 / len(row), axis=1)
Rows_Completeness
```

Then, all the rows completeness have been gotten.

```
Out[56]: 68516      90.0
        68517      90.0
        68518     100.0
        68519      80.0
        68520      90.0
        ...
        496058     100.0
        496059     100.0
        496060     100.0
        496061     100.0
        496062     100.0
        Length: 201125, dtype: float64
```

Get the mean of these completeness.

```
Rows_Completeness.mean()
```

```
Out[57]: 99.6997389683033
```

The result is 99.70 (2 Decimal)

The second one is:


```
# rows_empty stands for the row contains the empty cell.
# tmp.isnull().any(axis=1) can get the rows that are empty. axis = 1 stands
for low.
rows_empty = tmp.loc[tmp.isnull().any(axis=1)]

# len(tmp) - len(non_empty) Represents rows without any empty cells.
Another_Rows_Completeness = (len(tmp) - len(non_empty)) * 100 / len(tmp)
Another_Rows_Completeness
```

The answer is :

```
Out[44]: 98.03206960845246
```

The values obtained by both methods are very high, close to one hundred. We could infer that the completeness of the data is high. Besides, according to the formula in task 5 and task 7, it is not hard to find that if we divided these formulae by 100, we would get a percentage, which is the proportion of non-empty values in the total data.

I think this might be a good way to measure the completeness of the data. (Actually, I think divided by 100 change it to percentage will be more intuitive.)

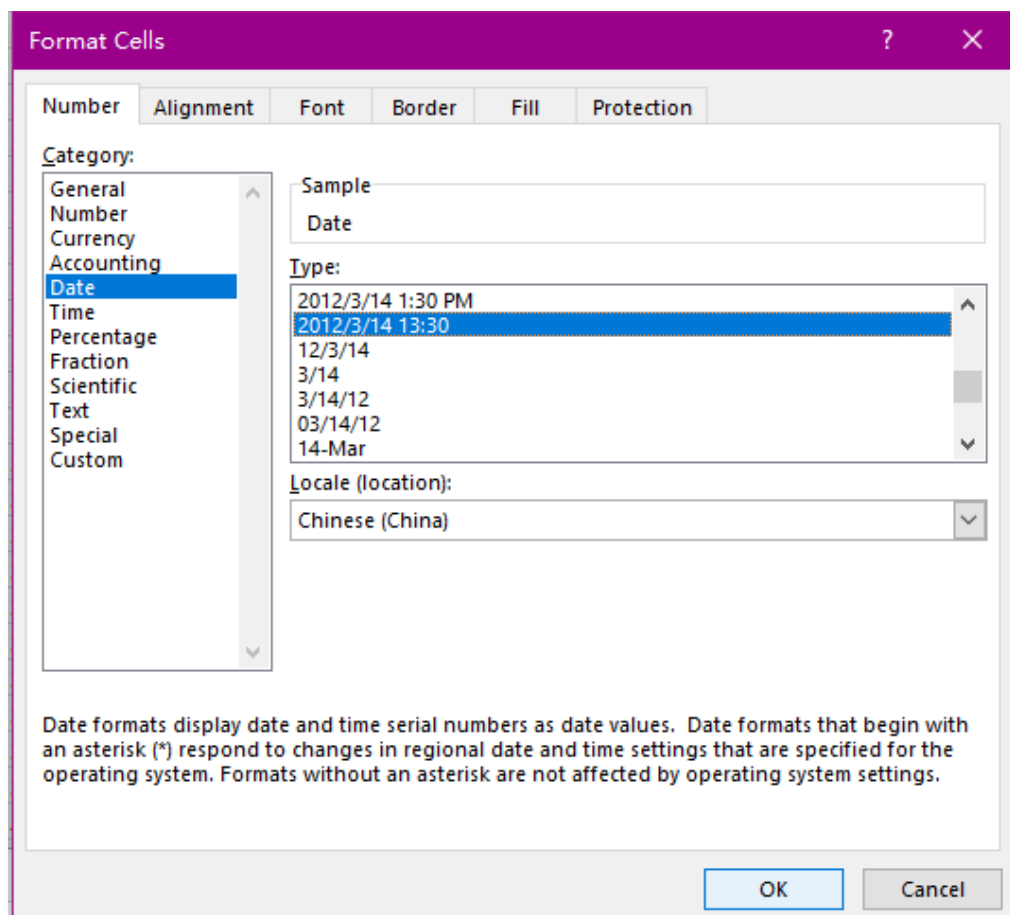
- **part2:**

I use Excel as my second tool to do task 6.

- **The first step:** calculate the average speed at 17:00 on Fridays on each lane.

1. Open the file 'rawpvr_2018-02-01_28d_1083 TueFri.csv', and the file rawpvr_2018-02-01_28d_1415 TueFri
2. Change the format of the date cell.

In file 1013, select the cell, right-click and choose format cell.





Then, the result is as following:

A	B	C	D	E	F	G	H	I	J
Date	Lane	Lane Name	Direction	Direction N	Speed (mp	Headway (Gap (s)	Flags	Flag Text
2018/2/2 0:00		6 SB_NS	2 South		38.525			0	
2018/2/2 0:00		5 SB_MID	2 South		32.31			0	
2018/2/2 0:00		4 SB_OS	2 South		44.739			0	
2018/2/2 0:00		6 SB_NS	2 South		33.554			0	
2018/2/2 0:00		6 SB_NS	2 South		39.768	12.3	11.847	0	
2018/2/2 0:00		2 NB_MID	1 North		64.623			0	
2018/2/2 0:00		1 NB_NS	1 North		29.205	6.319		0	
2018/2/2 0:00		2 NB_MID	1 North		37.283	6.2	6.089	0	
2018/2/2 0:01		6 SB_NS	2 South		44.739	14.8	14.575	0	
2018/2/2 0:01		2 NB_MID	1 North		41.01	5.155	5.242	0	
2018/2/2 0:01		2 NB_MID	1 North		37.283	1.47	0.949	0	
2018/2/2 0:01		5 SB_MID	2 South		36.039	47.1	47.017	0	
2018/2/2 0:01		6 SB_NS	2 South		36.661	12.3	12.24	0	
2018/2/2 0:01		3 NB_OS	1 North		45.361			0	
2018/2/2 0:01		2 NB_MID	1 North		38.525	41.3	41.06	0	
2018/2/2 0:01		5 SB_MID	2 South		47.224	38.9	38.639	0	
2018/2/2 0:01		6 SB_NS	2 South		57.787	35.7	35.438	0	
2018/2/2 0:01		6 SB_NS	2 South		47.846	4.301	3.334	0	
2018/2/2 0:01		1 NB_NS	1 North		44.117	61.4	61.086	0	
2018/2/2 0:01		6 SB_NS	2 South		49.709	1.957	1.599	0	

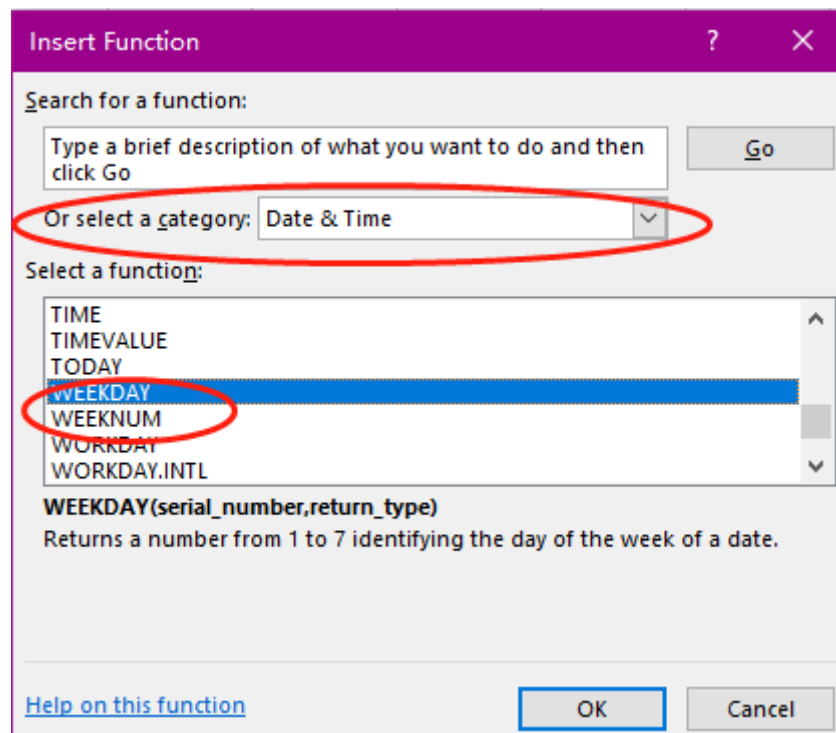
Use the same method to process files 1415.

A	B	C	D	E	F	G	H	I	J
Date	Lane	Lane Name	Direction	Direction N	Speed (mp	Headway (Gap (s)	Flags	Flag Text
2018/2/2 0:00		3 SW	2 SouthWest		26.098			0	
2018/2/2 0:00		3 SW	2 SouthWest		34.176	1.636	1.171	0	
2018/2/2 0:00		3 SW	2 SouthWest		24.855			0	
2018/2/2 0:00		3 SW	2 SouthWest		36.661	2.38	2.523	0	
2018/2/2 0:00		2 NE_OS	1 NorthEast		16.155			0	
2018/2/2 0:00		3 SW	2 SouthWest		20.506	6.6	6.307	0	
2018/2/2 0:00		2 NE_OS	1 NorthEast		44.739	4.6	11.346	0	
2018/2/2 0:00		3 SW	2 SouthWest		37.903	4.928	8.02	0	
2018/2/2 0:01		3 SW	2 SouthWest		39.146	9.3	8.964	0	
2018/2/2 0:01		2 NE_OS	1 NorthEast		22.991	13.5	13.265	0	
2018/2/2 0:01		3 SW	2 SouthWest		39.768	2.475	1.914	0	
2018/2/2 0:01		3 SW	2 SouthWest		50.331	8.4	8.124	0	
2018/2/2 0:01		3 SW	2 SouthWest		30.447	7.9	7.624	0	
2018/2/2 0:01		3 SW	2 SouthWest		32.31	2.319	1.567	0	
2018/2/2 0:01		3 SW	2 SouthWest		32.31	4.154	3.285	0	
2018/2/2 0:01		1 NE_NS	1 NorthEast		29.205			0	
2018/2/2 0:01		2 NE_OS	1 NorthEast		35.417	23.4	22.991	0	
2018/2/2 0:01		3 SW	2 SouthWest		42.253	1.271	0.638	0	
2018/2/2 0:01		2 NE_OS	1 NorthEast		17.399	10.929	4.728	0	
2018/2/2 0:01		3 SW	2 SouthWest		43.495	4.757	5.809	0	
2018/2/2 0:01		2 NE_OS	1 NorthEast		39.768	4.866	9.696	0	
2018/2/2 0:01		2 NE_OS	1 NorthEast		29.205	7.1	6.853	0	
2018/2/2 0:02		2 NE_OS	1 NorthEast		44.117	8	7.671	0	
2018/2/2 0:02		1 NE_NS	1 NorthEast		34.176	35.9	35.808	0	
2018/2/2 0:02		1 NE_NS	1 NorthEast		34.176	3.404	2.679	0	
2018/2/2 0:02		1 NE_NS	1 NorthEast		27.962	7	6.738	0	
2018/2/2 0:02		3 SW	2 SouthWest		44.117	46.1	45.878	0	
2018/2/2 0:02		2 NE_OS	1 NorthEast		36.039	22.6	22.346	0	
2018/2/2 0:02		2 NE_OS	1 NorthEast		40.39	3.046	2.802	0	
2018/2/2 0:02		3 SW	2 SouthWest		35.417	26.5	26.241	0	
2018/2/2 0:02		3 SW	2 SouthWest		34.798	2.25	1.503	0	
2018/2/2 0:03		2 NE_OS	1 NorthEast		24.855	40	39.729	0	

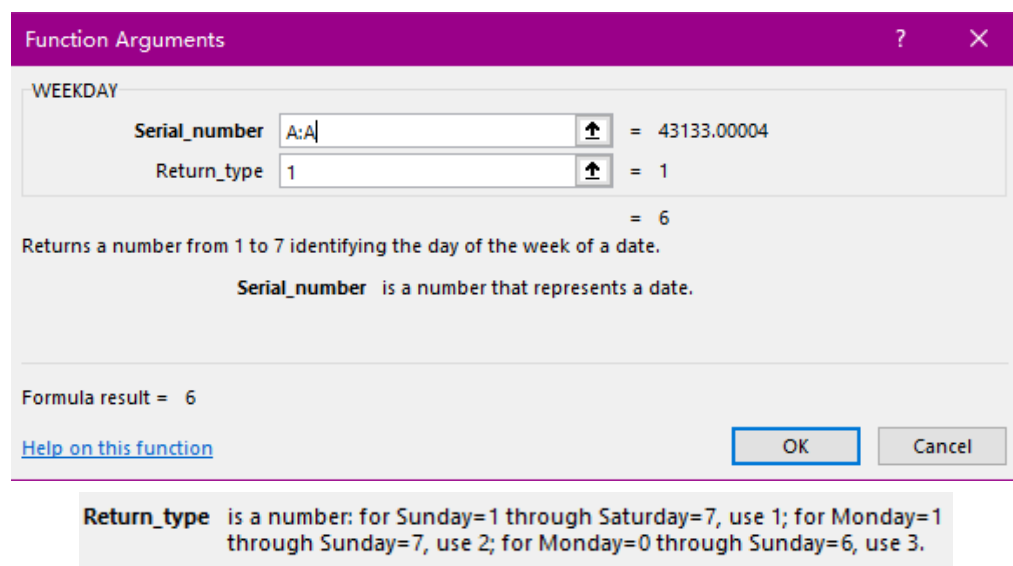
3. Use function WEEKDAY to calculate the label 'Flags.' Select the cell that needs to be deal with, then click  to insert a function.

<	✓		0				
B	C	D	E	F	G	H	I
Lane	Lane Name	Direction	Direction N	Speed (mp	Headway (Gap (s)	Flags
6	SB_NS	2	South	38.525			0
5	SB_MID	2	South	32.31			0
4	SB_OS	2	South	44.739			0

Select category Date & Time, and select WEEKDAY.



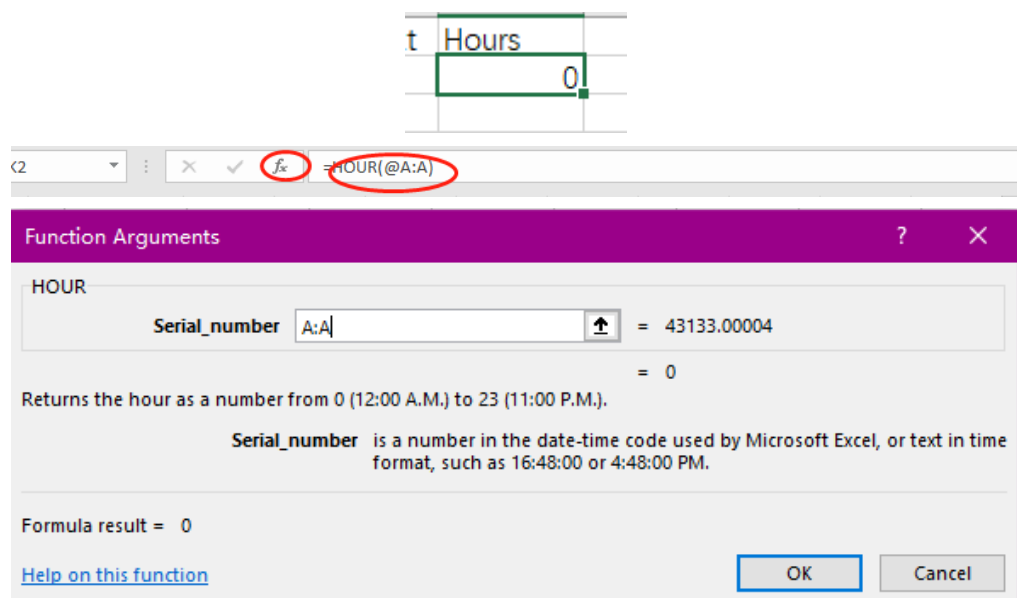
Then, select the Serials number and return type. According to the requirements of this task, 2 should be selected as the return type.



The label 'Flags' has been gotten, and the function WEEKDAY can be applied to the whole column. First, select the cell just be mentioned, right-click to copy. Second, Select the starting cell, press shift, and then select the ending cell.

2018/2/9 23:58	5 SB_MID	2 South	50.331	6.9	6.658	5
2018/2/9 23:58	5 SB_MID	2 South	36.661	7.9	7.731	5
2018/2/9 23:59	3 NB_OS	1 North	29.825	43.4	43.127	5
2018/2/9 23:59	5 SB_MID	2 South	41.632	12.9	12.644	5
2018/2/9 23:59	6 SB_NS	2 South	71.457	39.8	39.501	5
2018/2/9 23:59	3 NB_OS	1 North	39.146	14.5	14.177	5
2018/2/9 23:59	2 NB_MID	1 North	33.554	40.8	40.543	5
2018/2/9 23:59	5 SB_MID	2 South	34.176	14.7	14.48	5
2018/2/9 23:59	1 NB_NS	1 North	32.31	37.9	37.609	5
2018/2/9 23:59	5 SB_MID	2 South	54.68	2.414	3.373	5
2018/2/9 23:59	4 SB_OS	2 South	29.205	52.5	52.189	5
2018/2/9 23:59	4 SB_OS	2 South	27.962	2.16	1.632	5
2018/2/9 23:59	6 SB_NS	2 South	41.632	16.8	16.706	5
2018/2/9 23:59	2 NB_MID	1 North	39.768	13.8	13.533	5
2018/2/9 23:59	2 NB_MID	1 North	36.661	3.386	2.741	5
2018/2/9 23:59	1 NB_NS	1 North	29.825	21.2	20.771	5
2018/2/9 23:59	1 NB_NS	1 North	26.718	2.009	1.192	5
2018/2/9 23:59	2 NB_MID	1 North	45.982	12.6	12.338	5
2018/2/9 23:59	3 NB_OS	1 North	47.224	32.9	32.637	5
2018/2/9 23:59	6 SB_NS	2 South	37.283	24.8	24.558	5
2018/2/9 23:59	3 NB_OS	1 North	37.903	5.2	4.968	5
2018/2/9 23:59	6 SB_NS	2 South	37.283	2.94	2.348	5
2018/2/13 0:00	6 SB_NS	2 South	36.661			2
2018/2/13 0:00	5 SB_MID	2 South	50.331			2
2018/2/13 0:00	5 SB_MID	2 South	37.903			2
2018/2/13 0:00	1 NB_NS	1 North	27.34			2
2018/2/13 0:00	6 SB_NS	2 South	53.438			2
2018/2/13 0:01	1 NB_NS	1 North	31.069	30.7	30.356	2
2018/2/13 0:01	5 SB_MID	2 South	50.331	53.9	53.658	2
2018/2/13 0:01	6 SB_NS	2 South	43.495	48.4	48.044	2
2018/2/13 0:01	5 SB_MID	2 South	31.691	16.6	16.431	2
2018/2/13 0:01	6 SB_NS	2 South	30.447	18	17.825	2
2018/2/13 0:01	5 SB_MID	2 South	41.632	7.2	6.918	2
2018/2/13 0:01	5 SB_MID	2 South	41.632	11	10.78	2
2018/2/13 0:02	6 SB_NS	2 South	36.661	27	26.684	2
2018/2/13 0:02	2 NB_MID	1 North	34.798			2
2018/2/13 0:02	6 SB_NS	2 South	45.982	47.8	47.495	2
2018/2/13 0:02	4 SB_OS	2 South	49.709			2
2018/2/13 0:03	6 SB_NS	2 South	53.438	4.102	4.471	2
2018/2/13 0:03	6 SB_NS	2 South	36.661	9.6	9.412	2

4. Create the label 'Hours' to get the hours of a day. In the same way, the label 'Hours' could be gotten.



The screenshot shows an Excel spreadsheet with a cell containing the text 't Hours' and a value '0'. Below the spreadsheet, the formula bar shows the formula `=HOUR(@A:A)`. The 'Function Arguments' dialog box is open, showing the 'HOUR' function. The 'Serial_number' argument is set to 'A:A', and the result is displayed as '0'. The dialog box also includes a description of the function and a 'Help on this function' link.

The result is:

2018/2/16 9:59	4 SB_OS	2 South	35.417	3.411	3.03	5		9
2018/2/16 9:59	1 NB_NS	1 North	26.098	1.114	0.423	5		9
2018/2/16 9:59	6 SB_NS	2 South	35.417	11.4	11.116	5		9
2018/2/16 9:59	1 NB_NS	1 North	26.718	1.842	1.357	5		9
2018/2/16 9:59	1 NB_NS	1 North	26.098	1.371	0.657	5		9
2018/2/16 9:59	5 SB_MID	2 South	31.691	7.7	7.395	5		9
2018/2/16 9:59	5 SB_MID	2 South	41.632	2.149	2.219	5		9
2018/2/16 9:59	2 NB_MID	1 North	18.64	1.26	30.302	5		9
2018/2/16 9:59	6 SB_NS	2 South	38.525	9.6	9.272	5		9
2018/2/16 9:59	5 SB_MID	2 South	44.117	8.7	8.485	5		9
2018/2/16 9:59	6 SB_NS	2 South	31.691	5.929	4.545	5		9
2018/2/16 9:59	6 SB_NS	2 South	32.31	1.488	1.018	5		9
2018/2/16 9:59	6 SB_NS	2 South	32.31	1.385	0.771	5		9
2018/2/16 9:59	6 SB_NS	2 South	35.417	1.547	1.123	5		9
2018/2/16 9:59	6 SB_NS	2 South	34.798	2.218	1.672	5		9
2018/2/16 9:59	3 NB_OS	1 North	42.875	36	35.673	5		9
2018/2/16 9:59	2 NB_MID	1 North	35.417	14.8	14.38	5		9
2018/2/16 9:59	6 SB_NS	2 South	30.447	6.429	5.104	5		9
2018/2/16 9:59	2 NB_MID	1 North	18.021	11.731	2.873	5		9
2018/2/16 10:00	5 SB_MID	2 South	27.34			5		10
2018/2/16 10:00	6 SB_NS	2 South	43.495			5		10
2018/2/16 10:00	4 SB_OS	2 South	34.798			5		10
2018/2/16 10:00	6 SB_NS	2 South	43.495	1.749	1.369	5		10
2018/2/16 10:00	4 SB_OS	2 South	29.825	1.95	1.117	5		10
2018/2/16 10:00	3 NB_OS	1 North	31.691			5		10
2018/2/16 10:00	1 NB_NS	1 North	27.962			5		10
2018/2/16 10:00	2 NB_MID	1 North	30.447			5		10
2018/2/16 10:00	4 SB_OS	2 South	27.962	1.64	0.992	5		10
2018/2/16 10:00	1 NB_NS	1 North	27.34	1.514	0.848	5		10
2018/2/16 10:00	5 SB_MID	2 South	29.205			5		10
2018/2/16 10:00	3 NB_OS	1 North	27.34	2.986	1.975	5		10
2018/2/16 10:00	3 NB_OS	1 North	25.476	1.976	1.256	5		10
2018/2/16 10:00	1 NB_NS	1 North	30.447	2.131	1.665	5		10
2018/2/16 10:00	2 NB_MID	1 North	32.31	3.877	3.484	5		10
2018/2/16 10:00	5 SB_MID	2 South	31.691	4.129	3.986	5		10
2018/2/16 10:00	1 NB_NS	1 North	29.825	2.512	1.855	5		10
2018/2/16 10:00	5 SB_MID	2 South	31.691	1.024	0.595	5		10
2018/2/16 10:00	4 SB_OS	2 South	38.525	4.674	5.76	5		10
2018/2/16 10:00	3 NB_OS	1 North	28.584	2.309	1.922	5		10
2018/2/16 10:00	2 NB_MID	1 North	30.447	3.563	2.667	5		10


The same method can be applied to the file 1415 to get the label 'Flags' and create the label 'Hours.'

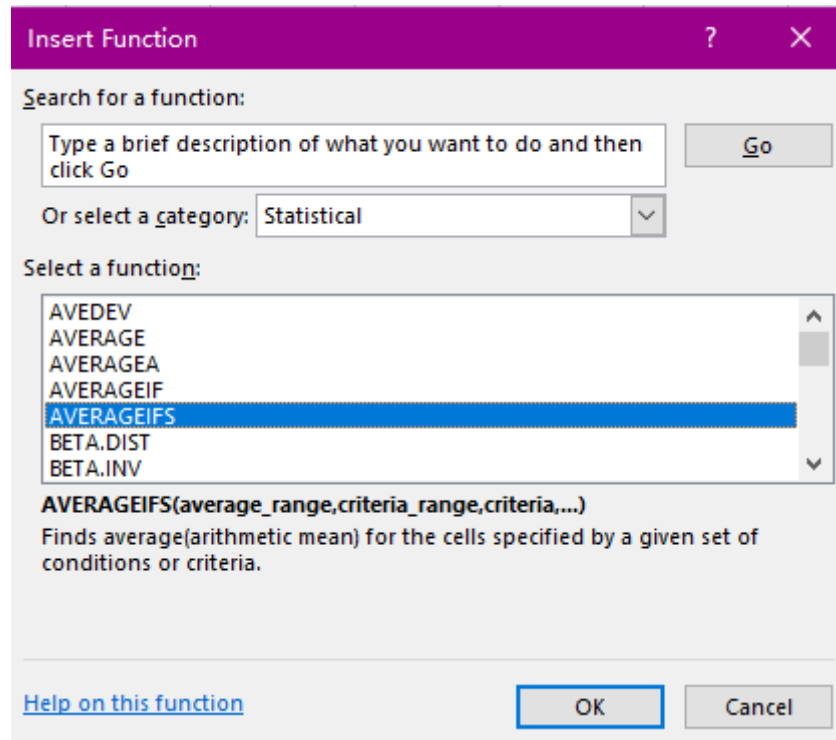
The result is as following:

Date	Lane	Lane Nam	Direction	Direction N	Speed (m	Headway (Gap (s)	Flags	Flag Text	Hours
2018/2/2 0:00	3 SW		2 SouthWest		26.098			5		0
2018/2/2 0:00	3 SW		2 SouthWest		34.176	1.636	1.171	5		0
2018/2/2 0:00	3 SW		2 SouthWest		24.855			5		0
2018/2/2 0:00	3 SW		2 SouthWest		36.661	2.38	2.523	5		0
2018/2/2 0:00	2 NE_OS		1 NorthEast		16.155			5		0
2018/2/2 0:00	3 SW		2 SouthWest		20.506	6.6	6.307	5		0
2018/2/2 0:00	2 NE_OS		1 NorthEast		44.739	4.6	11.346	5		0
2018/2/2 0:00	3 SW		2 SouthWest		37.903	4.928	8.02	5		0
2018/2/2 0:01	3 SW		2 SouthWest		39.146	9.3	8.964	5		0
2018/2/2 0:01	2 NE_OS		1 NorthEast		22.991	13.5	13.265	5		0
2018/2/2 0:01	3 SW		2 SouthWest		39.768	2.475	1.914	5		0
2018/2/2 0:01	3 SW		2 SouthWest		50.331	8.4	8.124	5		0
2018/2/2 0:01	3 SW		2 SouthWest		30.447	7.9	7.624	5		0
2018/2/2 0:01	3 SW		2 SouthWest		32.31	2.319	1.567	5		0
2018/2/2 0:01	3 SW		2 SouthWest		32.31	4.154	3.285	5		0
2018/2/2 0:01	1 NE_NS		1 NorthEast		29.205			5		0
2018/2/2 0:01	2 NE_OS		1 NorthEast		35.417	23.4	22.991	5		0
2018/2/2 0:01	3 SW		2 SouthWest		42.253	1.271	0.638	5		0
2018/2/2 0:01	2 NE_OS		1 NorthEast		17.399	10.929	4.728	5		0
2018/2/2 0:01	3 SW		2 SouthWest		43.495	4.757	5.809	5		0
2018/2/2 0:01	2 NE_OS		1 NorthEast		39.768	4.866	9.696	5		0
2018/2/2 0:01	2 NE_OS		1 NorthEast		29.205	7.1	6.853	5		0
2018/2/2 0:02	2 NE_OS		1 NorthEast		44.117	8	7.671	5		0
2018/2/2 0:02	1 NE_NS		1 NorthEast		34.176	35.9	35.808	5		0
2018/2/2 0:02	1 NE_NS		1 NorthEast		34.176	3.404	2.679	5		0
2018/2/2 0:02	1 NE_NS		1 NorthEast		27.962	7	6.738	5		0
2018/2/2 0:02	3 SW		2 SouthWest		44.117	46.1	45.878	5		0
2018/2/2 0:02	2 NE_OS		1 NorthEast		36.039	22.6	22.346	5		0
2018/2/2 0:02	2 NE_OS		1 NorthEast		40.39	3.046	2.802	5		0
2018/2/2 0:02	3 SW		2 SouthWest		35.417	26.5	26.241	5		0
2018/2/2 0:02	3 SW		2 SouthWest		34.798	2.25	1.503	5		0
2018/2/2 0:03	2 NE_OS		1 NorthEast		24.855	40	39.729	5		0
2018/2/2 0:03	3 SW		2 SouthWest		37.903	30.1	29.779	5		0
2018/2/2 0:03	3 SW		2 SouthWest		39.146	7.9	7.623	5		0
2018/2/2 0:03	3 SW		2 SouthWest		32.31	6.023	4.286	5		0
2018/2/2 0:03	3 SW		2 SouthWest		47.846	17.3	16.975	5		0
2018/2/2 0:03	2 NE_OS		1 NorthEast		35.417	47.3	46.895	5		0
2018/2/2 0:04	3 SW		2 SouthWest		34.176	12.6	12.329	5		0
2018/2/2 0:04	3 SW		2 SouthWest		41.01	8.1	7.825	5		0

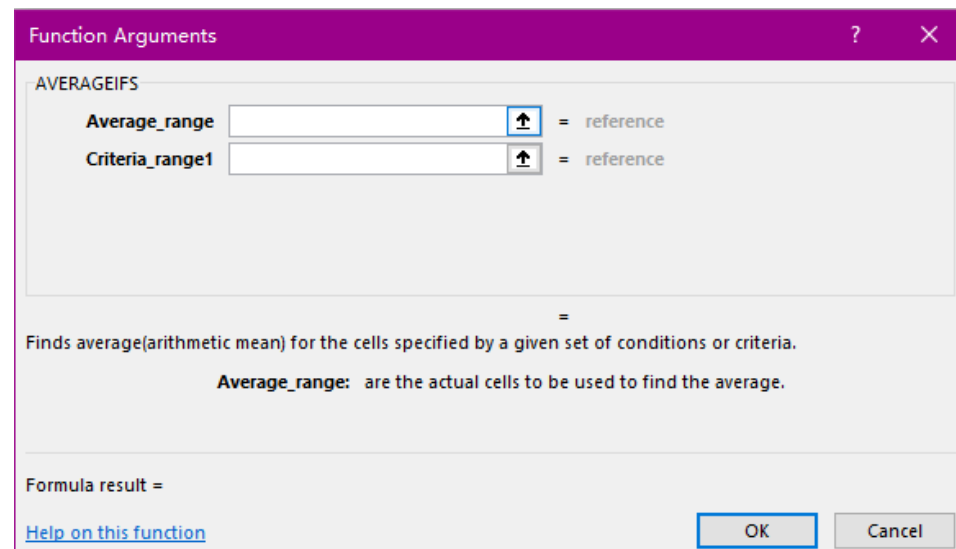
5. Calculate average speed by using the function AVERAGEIFS

1. Select a cell to put average.

2. Click  to use a function.
3. Find AVERAGEIFS, and click OK.



The following interface will be displayed:



'Speed' is used to find the average, so select speed.

	C	D	E	F	G	H	I	J
	Lane Name	Direction	Direction Name	Speed (mph)	Headway (Gap (s))	Flags	Flag Text	
6	SB_NS	2	South	38.525			0	
5	SB_MID	2	South	32.31			0	
4	SB_OS	2	South	44.739			0	
6	SB_NS	2	South	33.554			0	
6	SB_NS	2	South	39.768	12.3	11.847	0	
2	NB_MID	1	North	64.623			0	
1	NB_NS	1	North	29.205	6.319		0	
2	NB_MID	1	North	37.283	6.2	6.089	0	

Function Arguments

AVERAGEIFS

Average_range F:F = {"Speed (mph)";38.525;32.31;44.739;33

Criteria_range1 = reference

Formula result =

[Help on this function](#)

OK **Cancel**

It can be seen that F is 'Speed.'

- Get the average speed of each lane one by one.

Filter out the lane to the north in these two files. It can be seen that the lanes' number are 1, 2, 3 in file 1083 and 1, 2 in file 1415.

Date	Lane	Lane Name	Direction	Direction Name	Speed (mph)	Headway (Gap (s))	Flags	Flag Text	Hours
2018/2/2 0:00		NB_MID	1	North	64.623		5		0
2018/2/2 0:00		NB_MID	1	North	29.205	6.319	5		0
2018/2/2 0:00		NB_MID	1	North	37.283	6.2	5		0
2018/2/2 0:01		NB_MID	1	North	41.01	5.155	5		0
2018/2/2 0:01		NB_MID	1	North	37.283	1.47	5		0
2018/2/2 0:01		NB_OS	1	North	45.361		5		0
2018/2/2 0:01		NB_MID	1	North	38.525	41.3	5		0
2018/2/2 0:01		NB_NS	1	North	44.117	61.4	5		0
2018/2/2 0:01		NB_OS	1	North	39.146	17.7	5		0
2018/2/2 0:02		NB_MID	1	North	29.825	18.4	5		0
2018/2/2 0:02		NB_MID	1	North	28.584	53.2	5		0
2018/2/2 0:03		NB_MID	1	North	47.846	8.9	5		0
2018/2/2 0:03		NB_OS	1	North	41.632	1.639	5		0

It can be checked in the way that is shown below.

Direction

Sort

Sort Smallest to Largest

Sort Largest to Smallest

Sort by Color

Sheet View

Clear Filter From "Lane"

Filter by Color

Number Filters

Search

☒ (Select All)

☒ 1

☒ 2

OK **Cancel**

	A	B	C	D	E	F	G	H	I	J	K	L
	Date	Lane	Lane Name	Direction	Direction Name	Speed (mph)	Headway (Gap (s))	Flags	Flag Text	Hours		
1			1	NorthEast	16.155			5		0		
2			1	NorthEast	44.739	4.6	11.346	5		0		
3			1	NorthEast	22.991	13.5	13.265	5		0		
4			1	NorthEast	29.205			5		0		
5			1	NorthEast	35.417	23.4	22.991	5		0		
6			1	NorthEast	17.399	10.929	4.728	5		0		
7			1	NorthEast	39.768	4.866	9.696	5		0		
8			1	NorthEast	29.205	7.1	6.853	5		0		
9			1	NorthEast	44.117	8	7.671	5		0		
10			1	NorthEast	34.176	35.9	35.808	5		0		
11			1	NorthEast	34.176	3.404	2.679	5		0		
12			1	NorthEast	27.962	7	6.738	5		0		
13			1	NorthEast	36.039	22.6	22.346	5		0		
14			1	NorthEast	40.39	3.046	2.802	5		0		
15			1	NorthEast	24.855	40	39.729	5		0		
16			1	NorthEast	35.417	47.3	46.895	5		0		
17			1	NorthEast	33.554	29.6	29.328	5		0		
18			1	NorthEast	37.903	5.223	5.18	5		0		
19			1	NorthEast	16.155	11.7	4.293	5		0		
20			1	NorthEast	30.447	156.4	155.952	5		0		
21			1	NorthEast	35.417			5		0		
22			1	NorthEast	28.584			5		0		
23			1	NorthEast	32.932	17.2	16.856	5		0		

Calculate the **lane 1** in **file 1083** first:

Select 'Direction Name.'

AVERAGEIFS

Criteria2	'=5'	= "=5"
Criteria_range3	K:K	= {"Hours";0;0;0;0;0;0;0;0;0;0;0;0;0;0;0}
Criteria3	"> =17"	= "> =17"
Criteria_range4	K:K	= {"Hours";0;0;0;0;0;0;0;0;0;0;0;0;0;0;0}
Criteria4	"< 18"	= "< 18"

= 24.8898668

Finds average(arithmetic mean) for the cells specified by a given set of conditions or criteria.

Criteria2: is the condition or criteria in the form of a number, expression, or text that defines which cells will be used to find the average.

Formula result = 24.8898668

[Help on this function](#)

OK **Cancel**

=AVERAGEIFS(F:F,E:E,"=North",I:I,"=5",K:K,">=17",K:K,"<18",B:B,"=1")

24.889871

28.78131

we can also get the average speed of **lane 1, 2** in **file 1415**

27.15932

Then, calculate the final average speed.

24.88987	
28.78131	
30.63937	27.11329
24.09657	
27.15932	

The **the average of the five lanes** is **27.11329**, and we need to convert it from mph into km/h by times 1.6.

It is **43.38126**.

The average is different, because of **accuracy issues**.

- **The second step:** get the average time

Using distance 4.86km divide the average speed and times 60 to get the average time(min.)

=4.85/N6 * 60

	L	M	N	O
)				
)				
)	24.88987			
)	28.78131			
)	30.63937	27.11329	43.38126	6.707965
)	24.09657			
)	27.15932			
)				
)				

Therefore the final answer is 6.707965, is different from task 5, might be a loss of accuracy.

Assessment of the two technologies

- **similarities:**

- Both Excel and python have many functions to deal with data.

For example, in this coursework, the required data can be filtered by python's loc method or achieved using Excel's COUNTIFS, or could use AVERAGEIFS.

Excel and python have many functions with similar functions.

- Both Excel and Python can deal with CSV file.

- **differences:**

- Excel

- Advantage:

1. Excel has a graphical interface. When processing data, people only need to use the mouse to select the cell that needs to be processed and follow the instructions to write the conditions. This means that even people who are not programmers can perform some complex operations on the data. For programmers, data selection becomes more intuitive.

For example, in task 6, we need to deal with the label 'Direction,' 'Hours,' and 'Flags.' Using Excel is more intuitive and convenient than python.

2. When people do not know what method to use to process data, sometimes people can get the answer directly through the Excel menu name.

For example, when I calculated the label 'Flags,' I did not know the WEEKDAY function, but through a series of clicks, I found it without searching online.

- Limitation:

1. The amount of data that Excel can handle is smaller than python, and the larger the data, the slower the calculation speed.
 2. Excel can only be used for windows and mac, not for Linux.
 3. Excel is hard to deal with the loop!
- Python
 - Advantage:
 1. People can code functions to solve the same problem.
When faced with complex tasks, python can write complex tasks in a function, which can be easily called. Excel can only repeat this process manually.
 2. Python is a cross-platform language, can also be used on Linux.
 3. Python has many modules about machine learning and deep learning, which is more convenient to use after python processing data.
 4. Python can integrate SQL statements, making it more convenient to process databases.
 5. Python can handle large amounts of data.
 - Limitation:
 1. Compared with Excel, python is harder to learn. The free and flexible syntax may produce many bugs that are difficult to fix.

In this task, I think excel is more challenging to deal with conditions than python. Besides, it's difficult for others to reproduce the results, people must read a lengthy document, and python only needs to run the code.