Building on the combined lane data from assignment 2 ...

In [1]:

```
import pandas

def harmonic_mean_speed(group) :
    # YOUR CODE HERE

def combine_lanes(group) :
    return pandas.Series([group.Volume.mean(), group.Occupancy.mean(),
harmonic_mean_speed(group)], index=['Volume', 'Occupancy', 'Speed'])

laned_data = # YOUR CODE HERE

combined_lanes = laned_data.groupby('Time_AEST').apply(combine_lanes)
```

Task A: What is the statistical correlation between Volume, Occupancy and Speed?

(2 marks)

Include (via Markdown) a discussion/explanation of the results

In [ ]:

```
# YOUR CODE HERE
```

Task B: Compute *"normal"* values for each hour of each day of the week

(4 marks)

We have seen in assignment 2 that the traffic load tends to vary based on the time of the day and the day of the week.

We are now interesting in identifying traffic **anomalies**, i.e. where there is a change relative to this typical traffic pattern.

Such anomalies are typically due to events such as roadworks or traffic accidents.

For our analysis we will focus on 1 hour time periods (for example between 8am and 9am or between 5pm and 6pm).

Before we can identify atypical traffic on a particular day, we need to determine what is *"normal"* for each hour of each day of the week.

In [3]:

```
# INSERT YOUR CODE HERE to compute the average Speed, Volume and Occupancy
for each hour of each day of the week
```

```
# Note: whenever we say "average", that should be interpretted as harmonic
mean for the Speed variable. The average Volume and Occupancy is the simple
arithmetic mean.

normal = # YOUR CODE HERE
normal
```

Out[3]:

Day	Hour	Volume	Occupancy	Speed
0	**0**	1.382407	0.590278	97.019531
	1	0.931944	0.413287	97.220427
	2	0.912963	0.431667	97.742398
	3	1.260648	0.611250	98.446251
	4	3.001389	1.634861	99.111525
...	**...**	...	...	...
6	**19**	7.920370	3.403241	97.274611
	20	6.728704	2.727176	96.970753
	21	5.268056	2.142361	97.010168
	22	4.159722	1.691898	97.207784
	23	2.421296	0.995000	97.184809

168 rows × 3 columns

Task C: Display the normal values of each variable in a tabular format (by hour of day and day of week)

(4 marks)

Regardless of how your normal dataframe is structured, we now wish to present that normal data to our manager in a nice tabular fashion for each variable where there is one row for each hour of the day and a column for each day of the week.

Your normal dataframe will likely store the day of the week as a number, but for the purposes of displaying the data to our manager we will instead substitute day names as our column labels.

Hint: don't construct a new dataframe element by element, instead take your existing normal data frame and filter, reshape or stack/unstack to create the desired tabular output.

Round all displayed values to 1 decimal place (but don't alter the underlying normal data).

Note: these new tables are only generated for display purposes, not for later use in anomoly computation (which should use your normal dataframe directly)

In [4]:

```
# The parameter column will be either 'Volume', 'Speed' or 'Occupancy'
(which should be column names in your normal data frame)
def normal_by_day_hour(column) :
    # YOUR CODE HERE
```

In [5]:

```
normal_by_day_hour('Volume')
```

Out[5]:

Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
0	1.4	1.5	1.6	1.8	2.0	2.7	2.7
1	0.9	1.0	1.1	1.2	1.3	1.6	1.7
2	0.9	1.1	1.1	1.2	1.2	1.3	1.2
3	1.3	1.4	1.4	1.4	1.6	1.3	1.2
4	3.0	3.2	3.1	3.3	3.4	2.1	1.6
5	10.0	10.8	10.6	10.7	10.5	4.8	2.8
6	16.2	17.4	16.7	17.6	16.7	7.5	4.9
7	18.7	20.2	19.5	20.5	20.1	11.1	7.4
8	17.6	18.7	18.1	19.3	19.1	15.7	11.8

Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
9	16.8	16.7	16.4	17.6	18.3	20.2	17.5
10	17.7	16.4	16.3	17.5	19.7	22.5	21.0
11	17.8	16.8	16.5	17.9	20.8	21.8	21.0
12	17.8	17.9	17.2	19.1	21.7	20.6	19.8
13	19.2	19.8	19.4	21.1	22.3	20.7	18.2
14	22.3	23.5	22.7	23.4	22.7	20.5	18.0
15	22.2	21.7	22.2	21.6	21.6	19.5	17.9
16	21.9	22.1	21.5	21.6	20.2	18.2	16.9
17	20.9	22.5	20.6	21.4	20.4	17.5	14.2
18	13.7	16.2	15.4	17.2	16.5	11.9	10.3
19	8.3	8.8	9.4	10.9	11.3	8.3	7.9
20	6.3	6.8	7.1	9.2	8.6	7.6	6.7
21	5.0	5.7	5.7	7.6	8.0	7.7	5.3
22	3.6	4.2	4.3	5.8	7.2	7.6	4.2
23	2.3	2.4	2.5	3.5	5.1	4.9	2.4

In [6]:

```
normal_by_day_hour('Occupancy')
```

Out[6]:

Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
0	0.6	0.7	0.7	0.8	0.9	1.1	1.2
1	0.4	0.5	0.6	0.6	0.6	0.7	0.7
2	0.4	0.5	0.5	0.6	0.6	0.6	0.7
3	0.6	0.7	0.7	0.7	0.8	0.6	0.5
4	1.6	1.5	1.4	1.5	1.6	0.9	0.7
5	4.5	4.9	4.7	4.8	4.7	2.1	1.4
6	7.3	7.8	7.6	7.9	7.5	3.2	2.0
7	8.4	9.3	8.9	9.4	9.1	4.6	3.0
8	8.2	8.5	8.4	8.8	8.7	6.5	4.8
9	7.8	7.9	7.8	8.2	8.6	8.6	7.2
10	8.3	7.9	7.9	8.3	9.3	10.3	8.8
11	8.3	8.0	7.9	8.5	10.6	15.6	8.7
12	8.3	8.5	8.1	9.0	11.8	15.4	8.2
13	9.1	9.3	9.1	10.0	13.2	10.2	7.5
14	10.5	11.8	11.9	13.2	20.1	8.8	7.4
15	13.8	20.3	18.7	18.4	24.7	8.3	7.4
16	15.0	22.0	19.8	24.0	26.2	7.7	7.0
17	15.2	21.3	20.2	24.1	18.9	7.3	5.9

Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
18	6.2	7.9	8.9	10.5	8.4	4.9	4.3
19	3.5	3.8	4.0	4.9	5.0	3.3	3.4
20	2.7	2.9	3.0	3.9	3.9	3.1	2.7
21	2.1	2.4	2.4	3.2	3.3	3.1	2.1
22	2.0	1.8	1.9	2.4	3.0	3.4	1.7
23	1.0	1.1	1.1	1.5	2.4	2.1	1.0

In [7]:

```
normal_by_day_hour('Speed')
```

Out[7]:

Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
0	97.0	97.9	97.9	98.2	98.4	97.6	97.2
1	97.2	96.5	97.9	98.1	98.5	97.6	97.2
2	97.7	95.7	98.5	98.4	98.7	97.9	97.1
3	98.4	96.9	98.9	99.1	98.9	98.0	97.6
4	99.1	98.8	99.1	99.2	99.1	98.6	97.5
5	98.8	98.6	98.9	98.8	99.0	98.5	97.7
6	96.7	96.6	96.7	96.7	97.0	98.4	98.4
7	95.1	94.8	94.3	95.0	95.2	98.0	98.4
8	95.1	95.1	94.4	95.1	95.1	97.2	98.0

Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
9	95.6	95.0	94.6	95.3	95.1	95.5	96.3
10	95.4	94.9	94.5	95.4	94.8	88.5	95.0
11	95.5	95.3	94.8	95.3	87.6	58.7	95.0
12	95.8	94.9	95.1	94.8	81.9	56.9	95.1
13	95.5	94.5	94.4	92.9	75.6	83.4	96.0
14	91.2	87.8	82.9	77.4	49.9	93.0	96.0
15	71.1	48.6	52.0	51.5	39.1	95.7	96.1
16	62.5	43.7	47.5	39.5	34.6	95.3	95.8
17	58.7	45.3	45.0	39.2	46.9	94.3	95.6
18	94.7	86.8	74.0	71.6	86.0	96.3	96.9
19	97.0	97.2	97.2	96.6	96.8	97.0	97.3
20	97.2	97.0	97.3	97.0	97.2	96.9	97.0
21	97.3	97.1	97.4	97.1	97.1	96.6	97.0
22	97.2	96.1	97.5	97.6	97.2	96.1	97.2
23	97.5	97.0	97.8	98.0	97.6	97.0	97.2

Task D: Join the data for each hour of the sample period with the corresponding normal data for that hour of day and day of week

(9 marks)

Next we will compute the average Speed, Volume and Occupancy for each hour in the sample period (which for our data file is 01-05-2021 to 30-06-2021)

We then join that data with the normal data for the corresponding hour of day and day of week.

We are then able to compute for each hour in the sample period the difference between the average and the normal average for each of our variables (Speed, Volume and Occupancy).

We will express these differences as percentage differences relative to the normal value. For example, if the average speed is 95.908, but the normal average speed is 97.641 then the Speed Difference as a percentage will be -1.77%

In [8]:

```
# YOUR CODE HERE

diff = # YOUR CODE HERE

# YOUR CODE HERE

diff
```

Out[8]:

	Date	Day	Hour	Volume	Occupancy	Speed	Volume_Normal	Occupancy_Normal	Speed_Normal	Speed_Difference	Volume_Difference	Occupancy_Difference
0	2021-05-01	5	0	2.620833	1.112500	95.908087	2.658796	1.096759	97.641756	-1.775541	-1.427825	1.435205
1	2021-05-01	5	1	1.729167	0.782500	95.849091	1.602778	0.671991	97.585573	-1.779445	7.885615	16.445057
2	2021-05-01	5	2	1.208333	0.532083	95.681310	1.274537	0.560463	97.855785	-2.222121	-5.194333	-5.063605
3	2021-05	5	3	1.304167	0.589583	95.484044	1.319444	0.585000	97.966059	-2.533545	-1.157895	0.783476

	Date	Day	Hour	Volume	Occupancy	Speed	Volume_Normal	Occupancy_Normal	Speed_Normal	Speed_Difference	Volume_Difference	Occupancy_Difference
	-01											
4	2021-05-01	5	4	2.000000	0.923333	96.384245	2.129630	0.938056	98.554054	-2.201644	-6.086957	-1.569440
...	...	...	...	...	...	...	...	...	...	...	...	...
1459	2021-06-30	2	19	4.454167	2.024583	98.588871	9.436574	4.043611	97.218504	1.409574	-52.798901	-49.931305
1460	2021-06-30	2	20	2.970833	1.337500	97.980642	7.113426	3.036991	97.334488	0.663850	-58.236251	-55.959695
1461	2021-06-30	2	21	2.383333	1.074583	97.712559	5.702315	2.401852	97.372890	0.348833	-58.204108	-55.260216
1462	2021-06-30	2	22	2.162500	0.959583	97.896579	4.349537	1.854074	97.544766	0.360668	-50.282065	-48.244606
1463	2021-06-30	2	23	1.608333	0.712500	98.191780	2.456481	1.069259	97.793255	0.407518	-34.526951	-33.365085

1464 rows × 12 columns

Task E: For each variable, identify which hours of the sample period had the largest variation compared to normal values

(3 marks)

To identify traffic incidents such as crashes, we now identify the outliers where the difference between the actual and normal values of our variables is greatest.

We don't know which of our variables (Speed, Volume or Occupancy) will give us the best indication of a traffic incident such as a crash, so we will examine outliers for each variable in turn.

It also may not be immediately obvious whether we are looking for large increases or large decreases in each of our variables in order to identify a traffic incident.

With respect to speed, we would expect a crash to result in a decrease in average Speed, but what about Volume and Occupancy? Would we expect them to be higher or lower than normal during a traffic jam?

In [9]:

```
# Create a function that will identify the outliers in our difference data
# The parameter column will be either 'Volume', 'Speed' or 'Occupancy'
(which should be column names in your normal data frame)
# The parameter ascending will be True if we wish to find the largest
increases or False if we wish to find the largest decreases
# The paremater count tells us how many outlier rows to return. For
example, a value of 5 would mean return the 5 rows with the largest
differences
def find_worse(column, ascending, count) :
    # YOUR CODE HERE
```

In [10]:

```
worst_speed = find_worse('Speed', True, 5) # find the 5 rows with the worst
speed differences
worst_speed
```

Out[10]:

	Date	Day	Hour	Volume	Occupancy	Speed	Volume_Normal	Occupancy_Normal	Speed_Normal	Speed_Difference	Volume_Difference	Occupancy_Difference
516	2021-05-22	5	12	14.891667	37.021250	18.676790	20.606019	15.443009	56.880174	-67.164676	-27.731470	139.728212
2555	2021-05	1	15	15.679167	38.884167	20.428037	21.697222	20.267269	48.568903	-57.940089	-27.736525	91.856967

	Date	Day	Hour	Volume	Occupancy	Speed	Volume_Normal	Occupancy_Normal	Speed_Normal	Speed_Difference	Volume_Difference	Occupancy_Difference
	-11											
618	2021-05-26	2	18	15.316667	22.497917	32.418639	15.392593	8.912130	73.952657	-56.162983	-0.493263	152.441533
1165	2021-06-18	4	13	18.500000	24.635000	35.694000	22.348958	13.168698	75.569461	-52.766635	-17.222093	87.072406
1164	2021-06-18	4	12	19.545833	22.274167	41.263005	21.676778	11.755126	81.916548	-49.627998	-9.830543	89.484720

In [ ]:

```
# Fixme: should ascending be True or False when trying to using Volume to
identify traffic incidents?
worst_volume = find_worse('Volume', ???, 4) # find the 4 rows with the
worst volume difference
worst_volume
```

In [ ]:

```
# Fixme: should ascending be True or False when trying to using Occupancy
to identify traffic incidents?
worst_occupancy = find_worse('Occupancy', ???, 3) # find the 3 rows with
the worst occupancy difference
worst_occupancy
```

Task F: Preparing data to visualize the hours that appear most atypical

(7 marks)

Having identified the hours of the sample period where we suspect there may have been a traffic incident, we now wish to examine precisely what happened during those hours in more detail.

For each of our variables (Speed, Occupancy and Volumne) we ultimately wish to create a plot showing the actual data during that hour compared to the normal averages for each minute of that hour.

Before we can plot, we need to extract and compute the relevant data for the hour when our potential incident occurred.

This time, when computing the normal average for each minute, we will exclude the date of the potential incident (otherwise data from during the incident may skew the averages).

We will create two functions, one to extract and compute the normal averages for each minute of the particular hour of day and day of week (excluding the specified date) and the other to compute the data for each minute of the particular hour on the specified date.

In [12]:

```
# Add extra "helper" functions here if you wish (to avoid repeating
yourself)

# YOUR CODE HERE

def get_normal_by_minute(hour_of_day, day_of_week, date) :
    # YOUR CODE HERE

def get_atypical_by_minute(hour_of_day, day_of_week, date) :
    # YOUR CODE HERE
```

In [13]:

```
get_normal_by_minute(17, 6, '2021-06-20')
```

Out[13]:

Minute	Volume	Occupancy	Speed
0	13.906250	5.775000	96.315998
1	13.281250	5.334375	97.718211
2	15.218750	6.256250	95.358363
3	13.906250	5.696875	96.379254
4	15.062500	6.240625	94.989736
5	14.000000	5.768750	96.555807

Minute	Volume	Occupancy	Speed
6	15.000000	6.153125	96.157902
7	14.000000	5.781250	96.115916
8	15.031250	6.090625	96.286917
9	14.187500	5.775000	96.196855
10	15.406250	6.281250	96.738603
11	14.625000	5.937500	96.778887
12	15.250000	6.375000	95.615882
13	13.500000	5.618750	95.467792
14	14.687500	6.109375	96.361499
15	13.928571	5.764286	95.263960
16	14.642857	6.103571	95.728745
17	14.593750	6.106250	95.507709
18	14.843750	6.206250	96.027129
19	15.312500	6.437500	94.261487
20	14.437500	5.946875	95.439802
21	15.625000	6.643750	94.089707
22	15.187500	6.396875	94.022217
23	14.125000	5.778125	95.449345

Minute	Volume	Occupancy	Speed
24	15.093750	6.446875	93.699338
25	13.718750	5.481250	96.693568
26	14.562500	6.025000	94.441852
27	15.562500	6.343750	95.897026
28	16.000000	6.796875	92.963251
29	15.312500	6.265625	94.656420
30	14.093750	5.896875	95.268814
31	15.500000	6.528125	94.447956
32	13.375000	5.571875	94.717504
33	13.937500	5.681250	96.000421
34	16.437500	6.840625	94.769885
35	14.562500	6.012500	95.372725
36	14.125000	5.846875	95.753522
37	13.531250	5.615625	96.270796
38	14.687500	6.021875	96.342253
39	13.156250	5.362500	96.044566
40	14.937500	6.103125	96.192382
41	14.406250	5.962500	96.151454

Minute	Volume	Occupancy	Speed
42	14.500000	6.000000	95.924147
43	13.031250	5.412500	96.383584
44	13.500000	5.562500	95.988994
45	13.125000	5.306250	95.960996
46	14.125000	5.715625	96.030808
47	12.156250	4.903125	96.291542
48	14.156250	5.865625	95.034414
49	13.906250	5.843750	95.458753
50	14.312500	6.162500	95.280737
51	13.531250	5.581250	95.854301
52	12.875000	5.334375	96.237897
53	12.437500	5.021875	95.655673
54	11.781250	4.865625	95.220626
55	13.718750	5.709375	95.381135
56	12.656250	5.103125	96.519197
57	12.343750	5.090625	97.606425
58	11.437500	4.731250	96.214142
59	13.093750	5.428125	95.037560

In [14]:

```
get_atypical_by_minute(17, 6, '2021-06-20')
```

Out[14]:

Minute	Volume	Occupancy	Speed
0	15.50	6.600	94.141045
1	17.75	7.025	95.521992
2	20.50	8.525	95.828146
3	15.00	6.050	97.685349
4	17.00	6.900	94.149706
5	17.25	7.050	94.724514
6	17.50	7.550	93.724095
7	18.25	7.825	95.043704
8	14.25	6.025	95.939452
9	21.50	8.900	93.279952
10	17.25	7.175	94.774279
11	11.50	4.775	96.168222
12	16.50	7.325	91.268147
13	18.75	7.525	95.439674
14	14.00	6.375	94.335341
15	18.25	7.800	95.786037
16	21.50	8.775	96.001213

Minute	Volume	Occupancy	Speed
17	11.00	4.600	97.551635
18	14.75	6.200	96.315929
19	13.25	5.200	97.106387
20	16.25	6.525	96.453641
21	15.25	6.075	95.596025
22	12.75	5.150	95.974847
23	11.75	5.375	93.834971
24	18.75	7.950	93.227041
25	20.00	8.475	93.218730
26	13.00	5.150	96.477733
27	17.25	6.850	95.286261
28	15.00	6.700	94.226961
29	13.00	5.200	97.566181
30	17.75	7.400	97.078772
31	9.75	3.825	97.656254
32	12.00	5.200	96.482982
33	18.75	8.200	92.383820
34	17.25	7.400	93.819566

Minute	Volume	Occupancy	Speed
35	11.25	4.550	96.718745
36	21.75	8.925	97.612608
37	16.50	7.100	94.568628
38	11.00	4.300	99.008739
39	14.25	5.775	96.246752
40	13.75	5.700	96.693720
41	11.25	4.550	95.394584
42	16.50	6.800	94.386383
43	9.00	3.925	96.897630
44	15.50	6.350	94.819022
45	10.00	3.900	98.666941
46	14.25	5.750	95.514154
47	11.00	4.375	97.699066
48	14.50	5.925	94.134967
49	11.00	4.625	97.021271
50	12.50	5.200	95.582388
51	15.25	6.625	94.596525
52	14.00	5.725	93.185407

Minute	Volume	Occupancy	Speed
53	14.00	5.575	97.836152
54	9.25	3.650	97.493153
55	12.75	5.075	96.318159
56	8.75	3.800	94.059802
57	11.50	4.675	95.397799
58	12.00	4.850	94.083004
59	12.00	5.050	94.774400

Task G: Visualize the hours that appear most atypical

(7 marks)

We can now use this data to plot the actual and the normal values for each of our variables, for each minute of the hour.

We will create 3 subplots, one for each variable (Speed, Occupancy and Volumne) (see https://stackoverflow.com/questions/22483588/how-to-plot-multiple-dataframes-in-subplots)

The title of the figure (above the 3 subplots) should be as below and show the day of week, date and hour of day (see https://stackoverflow.com/questions/7066121/how-to-set-a-single-main-title-above-all-the-subplots-with-pyplot).

Remember, don't repeat yourself!

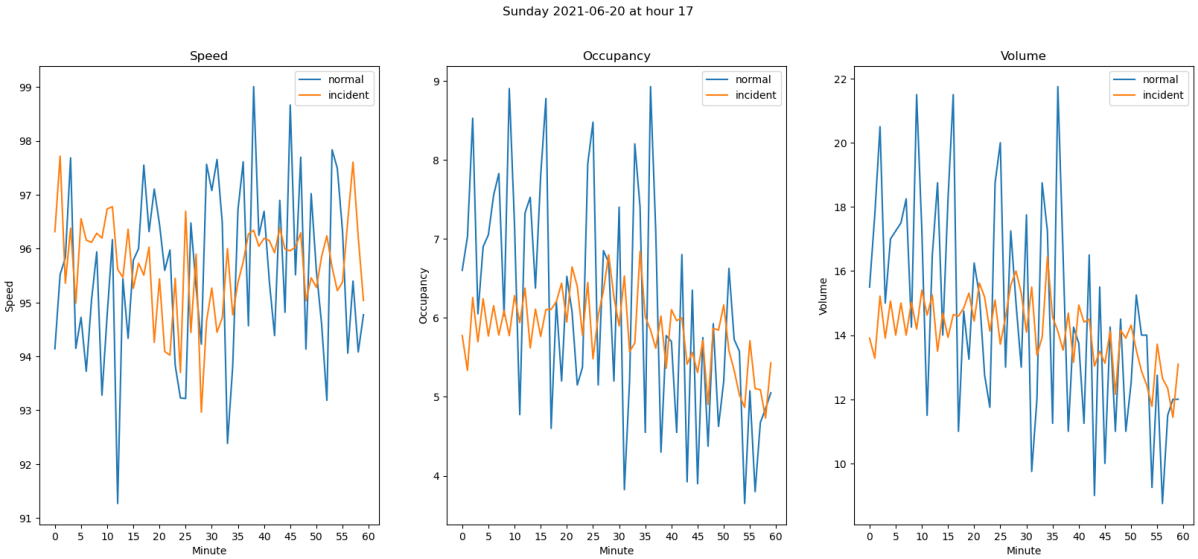
In [15]:

```
import matplotlib.pyplot

def plot_hour_comparision(hour_of_day, day_of_week, date) :
    variables = ['Speed', 'Occupancy', 'Volume']
    fig, axes = matplotlib.pyplot.subplots(ncols=len(variables),
figsize=(20,8))
    # YOUR CODE HERE
```

In [16]:

```
# Test example
plot_hour_comparision(17, 6, '2021-06-20')
```



Task H: Now that we can visualize a specific hour, we can do so for each of the rows in our outlier lists

(4 marks)

Normally we shouldn't use for loops to iterate over the rows in a Pandas data frame (we should use higher level Pandas operations instead).

However, this is one of the few circumstances where it is reasonable (see https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas)

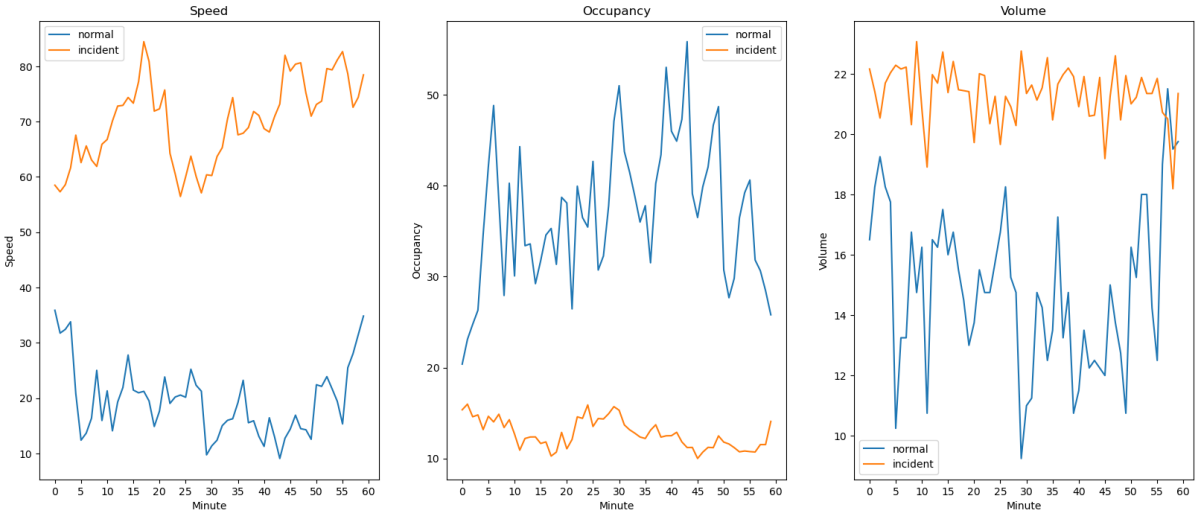
In [17]:

```
# The parameter worst will be data frame (such as worst_speed, worst_volume
or worse_occupancy) containing columns: Hour, Day and Date
# We don't know how many rows of data there will be, but we want to create
plots for each of the hours included in the list
def visualize_worst(worst) :
    # YOUR CODE HERE
```

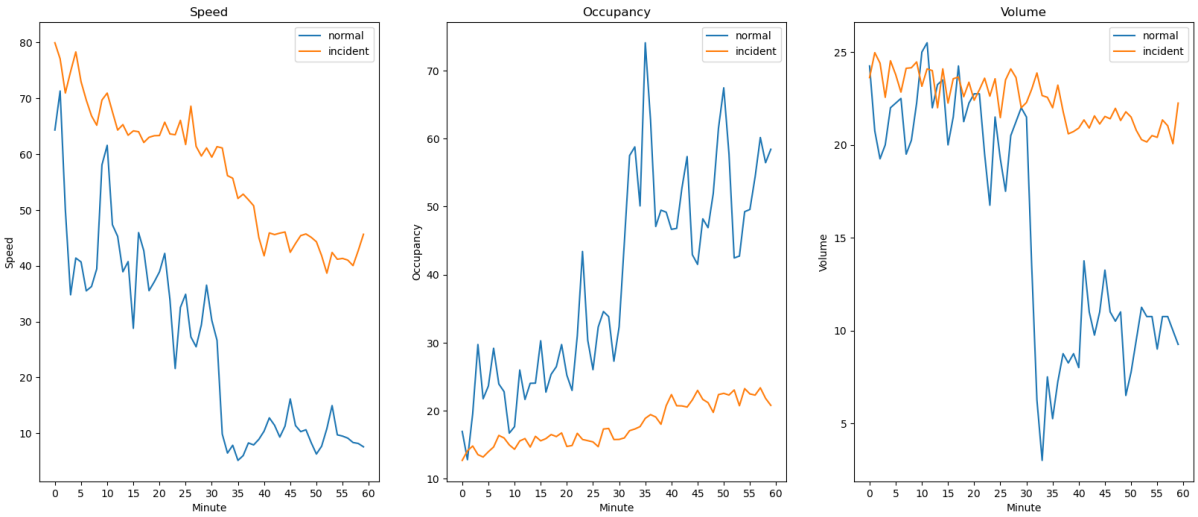
In [18]:

```
visualize_worst(worst_speed)
```

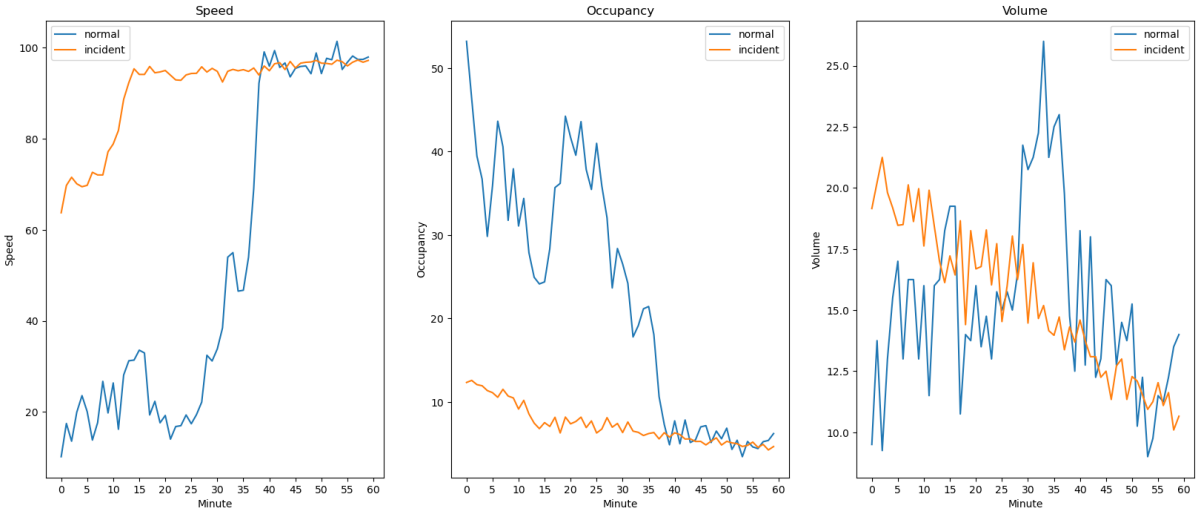
Saturday 2021-05-22 00:00:00 at hour 12

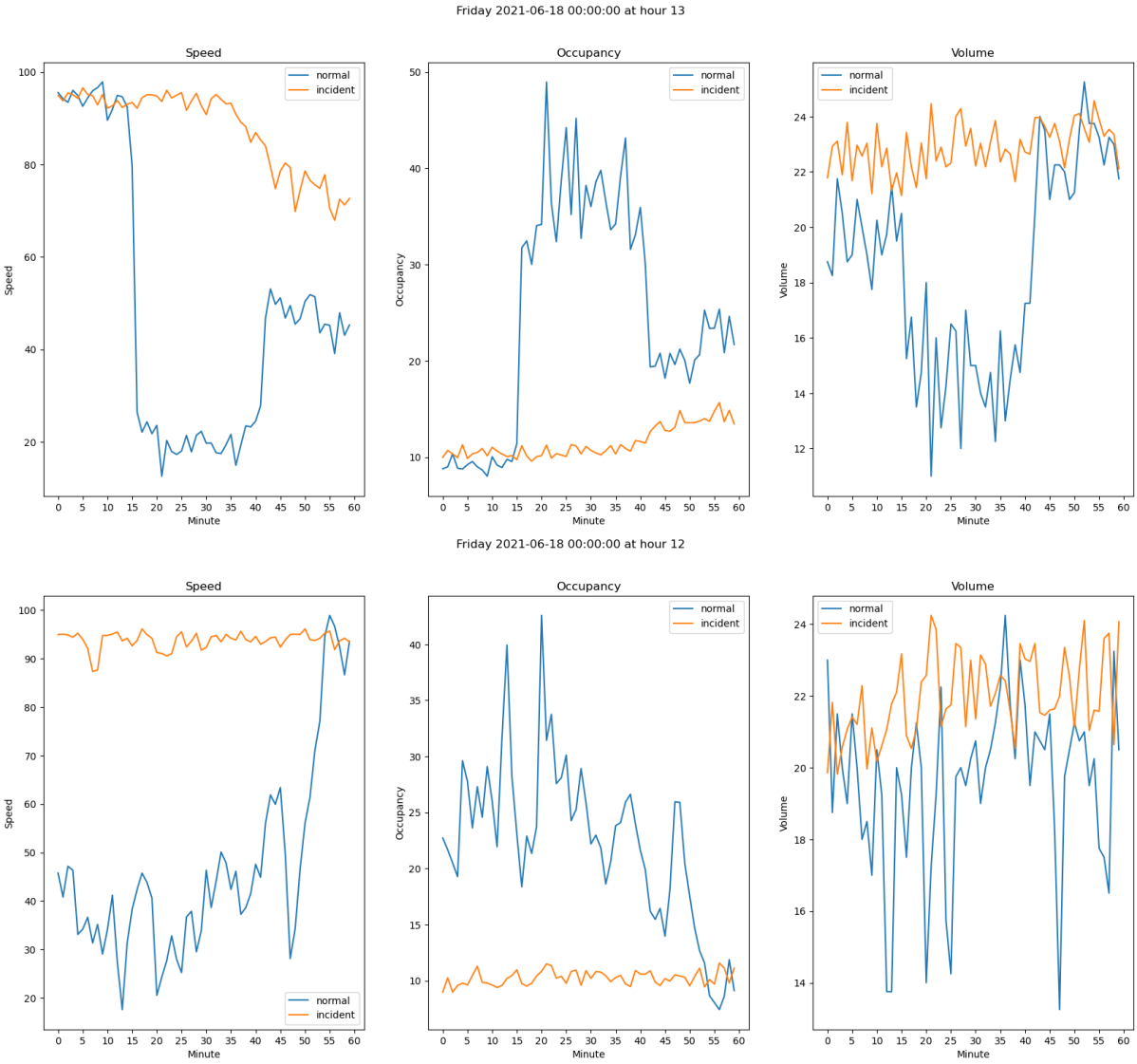


Tuesday 2021-05-11 00:00:00 at hour 15



Wednesday 2021-05-26 00:00:00 at hour 18





In [ ]:

```
visualize_worst(worst_volume)
```

In [ ]:

```
visualize_worst(worst_occupancy)
```

Which of our variables (Speed, Volume or Occupancy) seem to provide the most reliable indication of an actual traffic incident?

Please discuss below.