Reducing the number of high fatality accidents

📖 Background

You work for the road safety team within the department of transport and are looking into how they can reduce the number of major incidents. The safety team classes major incidents as fatal accidents involving 3+ casualties. They are trying to learn more about the characteristics of these major incidents so they can brainstorm interventions that could lower the number of deaths. They have asked for your assistance with answering a number of questions.

💾 The data

The reporting department have been collecting data on every accident that is reported. They've included this along with a lookup file for 2020's accidents.

Published by the department for transport. [https://data.gov.uk/dataset/road-accidents-safety-data (https://data.gov.uk/dataset/road-accidents-safety-data)](https://data.gov.uk/dataset/road-accidents-safety-data) Contains public sector information licensed under the Open Government Licence v3.0.

In [38]:

```
1  import pandas as pd
2  accidents = pd.read_csv(r'./data/accident-data.csv')
3  accidents.head()
```

Out[38]:

	accident_index	accident_year	accident_reference	longitude	latitude	accident_severity	number_of_vehicles	number_of_casualties	date
0	2020010219808	2020	10219808	-0.254001	51.462262	3	1	1	04/02/2020
1	2020010220496	2020	10220496	-0.139253	51.470327	3	1	2	27/04/2020
2	2020010228005	2020	10228005	-0.178719	51.529614	3	1	1	01/01/2020
3	2020010228006	2020	10228006	-0.001683	51.541210	2	1	1	01/01/2020
4	2020010228011	2020	10228011	-0.137592	51.515704	3	1	2	01/01/2020

5 rows × 27 columns

In [39]:

```
1  accidents.isnull()
```

Out[39]:

	accident_index	accident_year	accident_reference	longitude	latitude	accident_severity	number_of_vehicles	number_of_casualties	date	day
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
91194	False	False	False	False	False	False	False	False	False	
91195	False	False	False	False	False	False	False	False	False	
91196	False	False	False	False	False	False	False	False	False	
91197	False	False	False	False	False	False	False	False	False	
91198	False	False	False	False	False	False	False	False	False	

91199 rows × 27 columns

In [40]:

```python
accidents.isnull().sum()
```

Out[40]:

```
accident_index                              0
accident_year                               0
accident_reference                          0
longitude                                  14
latitude                                   14
accident_severity                           0
number_of_vehicles                          0
number_of_casualties                        0
date                                        0
day_of_week                                 0
time                                        0
first_road_class                            0
first_road_number                           0
road_type                                   0
speed_limit                                 0
junction_detail                             0
junction_control                            0
second_road_class                           0
second_road_number                          0
pedestrian_crossing_human_control           0
pedestrian_crossing_physical_facilities     0
light_conditions                            0
weather_conditions                          0
road_surface_conditions                     0
special_conditions_at_site                  0
carriageway_hazards                         0
urban_or_rural_area                         0
dtype: int64
```

In [41]:

```python
accidents.isnull().sum().sum()
```

Out[41]:

```
28
```

In [42]:

```python
accidents.date
```

Out[42]:

```
0         04/02/2020
1         27/04/2020
2         01/01/2020
3         01/01/2020
4         01/01/2020
             ...
91194     12/08/2020
91195     13/11/2020
91196     15/04/2020
91197     15/12/2020
91198     25/08/2020
Name: date, Length: 91199, dtype: object
```

In [43]:

```python
accidents1 = accidents.dropna()
accidents1
```

Out[43]:

	accident_index	accident_year	accident_reference	longitude	latitude	accident_severity	number_of_vehicles	number_of_casualties	da
0	2020010219808	2020	10219808	-0.254001	51.462262	3	1	1	04/02/20:
1	2020010220496	2020	10220496	-0.139253	51.470327	3	1	2	27/04/20:
2	2020010228005	2020	10228005	-0.178719	51.529614	3	1	1	01/01/20:
3	2020010228006	2020	10228006	-0.001683	51.541210	2	1	1	01/01/20:
4	2020010228011	2020	10228011	-0.137592	51.515704	3	1	2	01/01/20:
...	...	...	...	...	...	...	...	...	
91194	2020991027064	2020	991027064	-2.926320	56.473539	2	2	1	12/08/20:
91195	2020991029573	2020	991029573	-4.267565	55.802353	3	1	1	13/11/20:
91196	2020991030297	2020	991030297	-2.271903	57.186317	2	2	1	15/04/20:
91197	2020991030900	2020	991030900	-3.968753	55.950940	3	2	1	15/12/20:
91198	2020991032575	2020	991032575	-4.561040	56.003843	3	1	1	25/08/20:

91185 rows × 27 columns

In [44]:

```python
accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 91199 entries, 0 to 91198
Data columns (total 27 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   accident_index                           91199 non-null  object
 1   accident_year                            91199 non-null  int64
 2   accident_reference                       91199 non-null  object
 3   longitude                                91185 non-null  float64
 4   latitude                                 91185 non-null  float64
 5   accident_severity                        91199 non-null  int64
 6   number_of_vehicles                       91199 non-null  int64
 7   number_of_casualties                     91199 non-null  int64
 8   date                                     91199 non-null  object
 9   day_of_week                              91199 non-null  int64
 10  time                                     91199 non-null  object
 11  first_road_class                         91199 non-null  int64
 12  first_road_number                        91199 non-null  int64
 13  road_type                                91199 non-null  int64
 14  speed_limit                              91199 non-null  int64
 15  junction_detail                          91199 non-null  int64
 16  junction_control                         91199 non-null  int64
 17  second_road_class                        91199 non-null  int64
 18  second_road_number                       91199 non-null  int64
 19  pedestrian_crossing_human_control        91199 non-null  int64
 20  pedestrian_crossing_physical_facilities  91199 non-null  int64
 21  light_conditions                         91199 non-null  int64
 22  weather_conditions                       91199 non-null  int64
 23  road_surface_conditions                  91199 non-null  int64
 24  special_conditions_at_site               91199 non-null  int64
 25  carriageway_hazards                      91199 non-null  int64
 26  urban_or_rural_area                      91199 non-null  int64
dtypes: float64(2), int64(21), object(4)
memory usage: 18.8+ MB
```

In [45]:

```
accidents1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 91185 entries, 0 to 91198
Data columns (total 27 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   accident_index                          91185 non-null  object
 1   accident_year                           91185 non-null  int64
 2   accident_reference                      91185 non-null  object
 3   longitude                               91185 non-null  float64
 4   latitude                                91185 non-null  float64
 5   accident_severity                       91185 non-null  int64
 6   number_of_vehicles                      91185 non-null  int64
 7   number_of_casualties                    91185 non-null  int64
 8   date                                    91185 non-null  object
 9   day_of_week                             91185 non-null  int64
 10  time                                    91185 non-null  object
 11  first_road_class                        91185 non-null  int64
 12  first_road_number                       91185 non-null  int64
 13  road_type                               91185 non-null  int64
 14  speed_limit                             91185 non-null  int64
 15  junction_detail                         91185 non-null  int64
 16  junction_control                        91185 non-null  int64
 17  second_road_class                       91185 non-null  int64
 18  second_road_number                      91185 non-null  int64
 19  pedestrian_crossing_human_control       91185 non-null  int64
 20  pedestrian_crossing_physical_facilities 91185 non-null  int64
 21  light_conditions                        91185 non-null  int64
 22  weather_conditions                      91185 non-null  int64
 23  road_surface_conditions                 91185 non-null  int64
 24  special_conditions_at_site              91185 non-null  int64
 25  carriageway_hazards                     91185 non-null  int64
 26  urban_or_rural_area                     91185 non-null  int64
dtypes: float64(2), int64(21), object(4)
memory usage: 19.5+ MB
```

In [46]:

```
lookup = pd.read_csv(r'./data/road-safety-lookups.csv')
lookup.head()
```

Out[46]:

	table	field name	code/format	label	note
0	Accident	accident_index	NaN	NaN	unique value for each accident. The accident_i...
1	Accident	accident_year	NaN	NaN	NaN
2	Accident	accident_reference	NaN	NaN	In year id used by the police to reference a c...
3	Accident	longitude	NaN	NaN	Null if not known
4	Accident	Latitude	NaN	NaN	Null if not known

In [47]:

```
accidents.isnull().sum().sum()
```

Out[47]:

28

In [48]:

```
accidents1.duplicated().sum()
```

Out[48]:

0

In [49]:

```
from datetime import datetime
```

In [50]:

```
accidents1['date'] = pd.to_datetime(accidents1['date'])
```

In [51]:

```
accidents1.date.dtype
```

Out[51]:

dtype('<M8[ns]')

In [52]:

```
accidents1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 91185 entries, 0 to 91198
Data columns (total 27 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   accident_index                         91185 non-null  object
 1   accident_year                          91185 non-null  int64
 2   accident_reference                     91185 non-null  object
 3   longitude                              91185 non-null  float64
 4   latitude                               91185 non-null  float64
 5   accident_severity                      91185 non-null  int64
 6   number_of_vehicles                     91185 non-null  int64
 7   number_of_casualties                   91185 non-null  int64
 8   date                                   91185 non-null  datetime64[ns]
 9   day_of_week                            91185 non-null  int64
 10  time                                   91185 non-null  object
 11  first_road_class                       91185 non-null  int64
 12  first_road_number                      91185 non-null  int64
 13  road_type                              91185 non-null  int64
 14  speed_limit                            91185 non-null  int64
 15  junction_detail                        91185 non-null  int64
 16  junction_control                       91185 non-null  int64
 17  second_road_class                      91185 non-null  int64
 18  second_road_number                     91185 non-null  int64
 19  pedestrian_crossing_human_control      91185 non-null  int64
 20  pedestrian_crossing_physical_facilities  91185 non-null  int64
 21  light_conditions                       91185 non-null  int64
 22  weather_conditions                     91185 non-null  int64
 23  road_surface_conditions                91185 non-null  int64
 24  special_conditions_at_site             91185 non-null  int64
 25  carriageway_hazards                    91185 non-null  int64
 26  urban_or_rural_area                    91185 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(21), object(3)
memory usage: 19.5+ MB
```

💪 Competition challenge

Create a report that covers the following:

1. What time of day and day of the week do most major incidents happen?
2. Are there any patterns in the time of day/ day of the week when major incidents occur?
3. What characteristics stand out in major incidents compared with other accidents?
4. On what areas would you recommend the planning team focus their brainstorming efforts to reduce major incidents?

Question 1

What time of day and day of the week do most major incidents happen?

In [53]:

```python
major_accidents = accidents1.loc[accidents1['number_of_casualties'] >= 3]
major_accidents.head()
```

Out[53]:

	accident_index	accident_year	accident_reference	longitude	latitude	accident_severity	number_of_vehicles	number_of_casualties	date	day
11	2020010228024	2020	10228024	-0.058620	51.660013	2	3	4	2020-01-01	
136	2020010228521	2020	10228521	-0.086520	51.616219	3	1	4	2020-03-01	
173	2020010228625	2020	10228625	-0.190729	51.459756	3	3	3	2020-05-01	
187	2020010228670	2020	10228670	0.185884	51.478366	3	2	3	2020-05-01	
212	2020010228789	2020	10228789	-0.251891	51.553577	2	2	4	2020-06-01	

5 rows × 27 columns

In [22]:

```
major_accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4817 entries, 11 to 91183
Data columns (total 27 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   accident_index                            4817 non-null   object
 1   accident_year                             4817 non-null   int64
 2   accident_reference                        4817 non-null   object
 3   longitude                                 4817 non-null   float64
 4   latitude                                  4817 non-null   float64
 5   accident_severity                         4817 non-null   int64
 6   number_of_vehicles                        4817 non-null   int64
 7   number_of_casualties                      4817 non-null   int64
 8   date                                      4817 non-null   datetime64[ns]
 9   day_of_week                               4817 non-null   int64
 10  time                                      4817 non-null   object
 11  first_road_class                          4817 non-null   int64
 12  first_road_number                         4817 non-null   int64
 13  road_type                                 4817 non-null   int64
 14  speed_limit                               4817 non-null   int64
 15  junction_detail                           4817 non-null   int64
 16  junction_control                          4817 non-null   int64
 17  second_road_class                         4817 non-null   int64
 18  second_road_number                        4817 non-null   int64
 19  pedestrian_crossing_human_control         4817 non-null   int64
 20  pedestrian_crossing_physical_facilities   4817 non-null   int64
 21  light_conditions                          4817 non-null   int64
 22  weather_conditions                        4817 non-null   int64
 23  road_surface_conditions                   4817 non-null   int64
 24  special_conditions_at_site                4817 non-null   int64
 25  carriageway_hazards                       4817 non-null   int64
 26  urban_or_rural_area                       4817 non-null   int64
dtypes: datetime64[ns](1), float64(2), int64(21), object(3)
memory usage: 1.0+ MB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4817 entries, 11 to 91183
Data columns (total 27 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   accident_index                            4817 non-null   object
 1   accident_year                             4817 non-null   int64
 2   accident_reference                        4817 non-null   object
 3   longitude                                 4817 non-null   float64
 4   latitude                                  4817 non-null   float64
 5   accident_severity                         4817 non-null   int64
 6   number_of_vehicles                        4817 non-null   int64
 7   number_of_casualties                      4817 non-null   int64
 8   date                                      4817 non-null   datetime64[ns]
 9   day_of_week                               4817 non-null   int64
 10  time                                      4817 non-null   object
 11  first_road_class                          4817 non-null   int64
 12  first_road_number                         4817 non-null   int64
 13  road_type                                 4817 non-null   int64
 14  speed_limit                               4817 non-null   int64
 15  junction_detail                           4817 non-null   int64
 16  junction_control                          4817 non-null   int64
 17  second_road_class                         4817 non-null   int64
 18  second_road_number                        4817 non-null   int64
 19  pedestrian_crossing_human_control         4817 non-null   int64
 20  pedestrian_crossing_physical_facilities   4817 non-null   int64
 21  light_conditions                          4817 non-null   int64
 22  weather_conditions                        4817 non-null   int64
 23  road_surface_conditions                   4817 non-null   int64
 24  special_conditions_at_site                4817 non-null   int64
 25  carriageway_hazards                       4817 non-null   int64
 26  urban_or_rural_area                       4817 non-null   int64
dtypes: datetime64[ns](1), float64(2), int64(21), object(3)
memory usage: 1.0+ MB
```

In [55]:

```
major_accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4817 entries, 11 to 91183
Data columns (total 27 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   accident_index                          4817 non-null   object
 1   accident_year                           4817 non-null   int64
 2   accident_reference                      4817 non-null   object
 3   longitude                               4817 non-null   float64
 4   latitude                                4817 non-null   float64
 5   accident_severity                       4817 non-null   int64
 6   number_of_vehicles                      4817 non-null   int64
 7   number_of_casualties                    4817 non-null   int64
 8   date                                    4817 non-null   datetime64[ns]
 9   day_of_week                             4817 non-null   int64
 10  time                                    4817 non-null   object
 11  first_road_class                        4817 non-null   int64
 12  first_road_number                       4817 non-null   int64
 13  road_type                               4817 non-null   int64
 14  speed_limit                             4817 non-null   int64
 15  junction_detail                         4817 non-null   int64
 16  junction_control                        4817 non-null   int64
 17  second_road_class                       4817 non-null   int64
 18  second_road_number                      4817 non-null   int64
 19  pedestrian_crossing_human_control       4817 non-null   int64
 20  pedestrian_crossing_physical_facilities 4817 non-null   int64
 21  light_conditions                        4817 non-null   int64
 22  weather_conditions                      4817 non-null   int64
 23  road_surface_conditions                 4817 non-null   int64
 24  special_conditions_at_site              4817 non-null   int64
 25  carriageway_hazards                     4817 non-null   int64
 26  urban_or_rural_area                     4817 non-null   int64
dtypes: datetime64[ns](1), float64(2), int64(21), object(3)
memory usage: 1.0+ MB
```

In [56]:

```
major_accidents.groupby('time')['time'].agg(['count']).sort_values(['count'], ascending=False)
```

Out[56]:
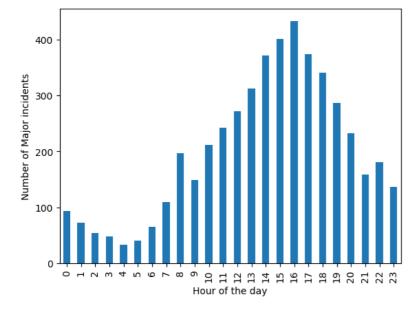
	count
time	
16:00	35
15:30	35
16:30	34
14:00	33
15:00	31
...	...
10:02	1
02:03	1
02:43	1
10:06	1
02:23	1

1132 rows × 1 columns

In [57]:

```python
import matplotlib.pyplot as plt
major_accidents.groupby(pd.to_datetime(major_accidents['time']).dt.hour)['time'].agg('count').plot(kind='bar')
plt.xlabel("Hour of the day");
plt.ylabel("Number of Major incidents");
```


In [58]:

```python
major_accidents.groupby('day_of_week')['day_of_week'].agg(['count']).sort_values(['count'], ascending=False)
```

Out[58]:
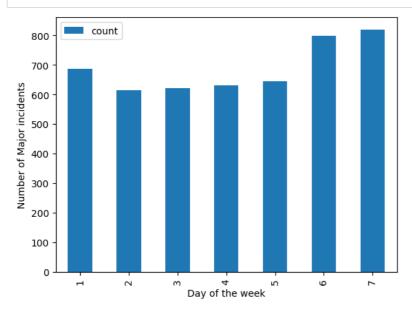
day_of_week	count
7	820
6	798
1	687
5	645
4	632
3	621
2	614

In [59]:

```python
major_accidents.groupby('day_of_week')['day_of_week'].agg(['count']).plot(kind='bar')
plt.xlabel("Day of the week");
plt.ylabel("Number of Major incidents");
```

Quest 2

Are there any patterns in the time of day/ day of the week when major incidents occur?

Discovery 1 (Patterns in time of day)
- From the chart of the time the major accidents occured, it is seen that the accidents with the highest number of casualties occured mostly between, 10am and 20pm(8pm) in the day.
- The highest number of major accidents occured at 16pm in the day with over 400 number of incidents.

Discovery 2 (Patterns in day of week)
- There is almost an even pattern of major accidents occurences in the days of the week.
- The highest number of accidents in the week happened on Friday and Saturday of the week.

Quest 3

What characteristics stand out in major incidents compared with other accidents?

In [60]:

```
other_accidents = accidents1.loc[accidents1['number_of_casualties'] <= 3]
other_accidents.head()
```

Out[60]:

	accident_index	accident_year	accident_reference	longitude	latitude	accident_severity	number_of_vehicles	number_of_casualties	date	day_c
0	2020010219808	2020	10219808	-0.254001	51.462262	3	1	1	2020-04-02	
1	2020010220496	2020	10220496	-0.139253	51.470327	3	1	2	2020-04-27	
2	2020010228005	2020	10228005	-0.178719	51.529614	3	1	1	2020-01-01	
3	2020010228006	2020	10228006	-0.001683	51.541210	2	1	1	2020-01-01	
4	2020010228011	2020	10228011	-0.137592	51.515704	3	1	2	2020-01-01	

5 rows × 27 columns

In [61]:

```python
other_accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 89582 entries, 0 to 91198
Data columns (total 27 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   accident_index                          89582 non-null  object
 1   accident_year                           89582 non-null  int64
 2   accident_reference                      89582 non-null  object
 3   longitude                               89582 non-null  float64
 4   latitude                                89582 non-null  float64
 5   accident_severity                       89582 non-null  int64
 6   number_of_vehicles                      89582 non-null  int64
 7   number_of_casualties                    89582 non-null  int64
 8   date                                    89582 non-null  datetime64[ns]
 9   day_of_week                             89582 non-null  int64
 10  time                                    89582 non-null  object
 11  first_road_class                        89582 non-null  int64
 12  first_road_number                       89582 non-null  int64
 13  road_type                               89582 non-null  int64
 14  speed_limit                             89582 non-null  int64
 15  junction_detail                         89582 non-null  int64
 16  junction_control                        89582 non-null  int64
 17  second_road_class                       89582 non-null  int64
 18  second_road_number                      89582 non-null  int64
 19  pedestrian_crossing_human_control       89582 non-null  int64
 20  pedestrian_crossing_physical_facilities 89582 non-null  int64
 21  light_conditions                        89582 non-null  int64
 22  weather_conditions                      89582 non-null  int64
 23  road_surface_conditions                 89582 non-null  int64
 24  special_conditions_at_site              89582 non-null  int64
 25  carriageway_hazards                     89582 non-null  int64
 26  urban_or_rural_area                     89582 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(21), object(3)
memory usage: 19.1+ MB
```

In [62]:

```python
other_accidents.groupby('time')['time'].agg(['count']).sort_values(['count'], ascending=False)
```

Out[62]:
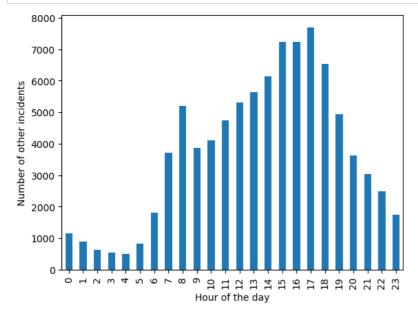
	count
time	
17:00	854
16:00	775
15:00	766
17:30	736
18:00	731
...	...
04:39	1
03:36	1
04:33	1
04:31	1
04:58	1

1438 rows × 1 columns

In [63]:

```python
other_accidents.groupby(pd.to_datetime(other_accidents['time']).dt.hour)['time'].agg('count').plot(kind='bar')
plt.xlabel("Hour of the day");
plt.ylabel("Number of other incidents");
```


In [64]:

```python
other_accidents.groupby('day_of_week')['day_of_week'].agg(['count']).sort_values(['count'], ascending=False)
```

Out[64]:
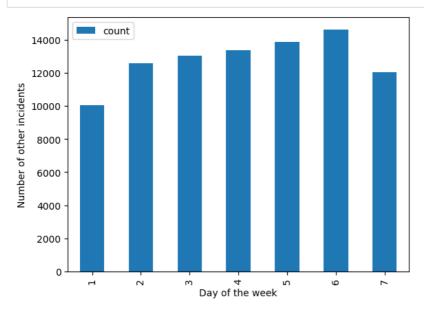
day_of_week	count
6	14629
5	13863
4	13360
3	13056
2	12579
7	12044
1	10051

In [65]:

```python
other_accidents.groupby('day_of_week')['day_of_week'].agg(['count']).plot(kind='bar')
plt.xlabel("Day of the week");
plt.ylabel("Number of other incidents");
```

Quest 3 continued...

- From the comparison of both the major accidents and the other accidents, it is seen that the major accidents occured mostly in the afternoon and evening hours with a few of them happening in the morning hours, while the other accidents was shown to occur mostly in the morning and mid-day hours of the day.
- Comparing the occurences of both accidents evaluation using the day of the week charts, it is seen that the highest number of major accidents happened on Saturday which is the seventh day of the week, while the highest number of other accidents happened Friday which is the sixth day of the weeek.

Question 4

On what areas would you recommend the planning team focus their brainstorming efforts to reduce major incidents?

From the analysis, majority of the both accidents occured during weekends and between the hours of 8am and 20pm, therefore I would recommend that the planning team focus their attention on discovering the causages of accidents between these hours and days mentioned and as well develop strategies to curb them.

In [66]:

```
accidents1
```

Out[66]:

	accident_index	accident_year	accident_reference	longitude	latitude	accident_severity	number_of_vehicles	number_of_casualties	date	d
0	2020010219808	2020	10219808	-0.254001	51.462262	3	1	1	2020-04-02	
1	2020010220496	2020	10220496	-0.139253	51.470327	3	1	2	2020-04-27	
2	2020010228005	2020	10228005	-0.178719	51.529614	3	1	1	2020-01-01	
3	2020010228006	2020	10228006	-0.001683	51.541210	2	1	1	2020-01-01	
4	2020010228011	2020	10228011	-0.137592	51.515704	3	1	2	2020-01-01	
...	...	...	...	...	...	...	...	...	...	
91194	2020991027064	2020	991027064	-2.926320	56.473539	2	2	1	2020-12-08	
91195	2020991029573	2020	991029573	-4.267565	55.802353	3	1	1	2020-11-13	
91196	2020991030297	2020	991030297	-2.271903	57.186317	2	2	1	2020-04-15	
91197	2020991030900	2020	991030900	-3.968753	55.950940	3	2	1	2020-12-15	
91198	2020991032575	2020	991032575	-4.561040	56.003843	3	1	1	2020-08-25	

91185 rows × 27 columns

Machine Learning

Drop the columns that are not needed for training the model or making predictions

In [67]:

```
accidents1.columns.unique()
```

Out[67]:

```
Index(['accident_index', 'accident_year', 'accident_reference', 'longitude',
       'latitude', 'accident_severity', 'number_of_vehicles',
       'number_of_casualties', 'date', 'day_of_week', 'time',
       'first_road_class', 'first_road_number', 'road_type', 'speed_limit',
       'junction_detail', 'junction_control', 'second_road_class',
       'second_road_number', 'pedestrian_crossing_human_control',
       'pedestrian_crossing_physical_facilities', 'light_conditions',
       'weather_conditions', 'road_surface_conditions',
       'special_conditions_at_site', 'carriageway_hazards',
       'urban_or_rural_area'],
      dtype='object')
```

In [68]:

```python
accidents1.drop(['pedestrian_crossing_human_control', 'pedestrian_crossing_physical_facilities',
                 'carriageway_hazards', 'urban_or_rural_area',
                 'second_road_number', 'first_road_number'],
                axis=1, inplace=True)
```

Split the dataset into y and X

To be further used during the logistic regression split function

'y' will serve as our targeted result while 'X' will serve as the prediction features

In [69]:

```python
y = accidents1['accident_severity']
```

In [70]:

```python
y
```

Out[70]:

```
0        3
1        3
2        3
3        2
4        3
        ..
91194    2
91195    3
91196    2
91197    3
91198    3
Name: accident_severity, Length: 91185, dtype: int64
```

In [72]:

```python
X = accidents1[['number_of_vehicles','number_of_casualties', 'speed_limit', 'weather_conditions',
                'light_conditions', 'road_surface_conditions']]
```

In [73]:

```python
X
```

Out[73]:

	number_of_vehicles	number_of_casualties	speed_limit	weather_conditions	light_conditions	road_surface_conditions
0	1	1	20	9	1	9
1	1	2	20	1	1	1
2	1	1	30	1	4	2
3	1	1	30	1	4	1
4	1	2	30	1	4	1
...	...	...	...	...	...	...
91194	2	1	30	1	1	1
91195	1	1	30	1	1	1
91196	2	1	60	1	1	1
91197	2	1	30	1	1	1
91198	1	1	30	1	1	1

91185 rows × 6 columns

Logistic Regression

Split the data set into the different sets needed for the logistic regression

In [74]:

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,train_size=0.75,random_state=101)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(68388, 6)
(22797, 6)
(68388,)
(22797,)
```

In [75]:

```python
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train,y_train)
```

Out[75]:

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Predict the x_test and x_train and check for the accuracy score

In [76]:

```python
y_test_pred = logreg.predict(x_test)
y_test_pred
```

Out[76]:

```
array([3, 3, 3, ..., 3, 3, 3])
```

In [77]:

```python
# Compare y_test and y_test_pred

from sklearn.metrics import accuracy_score
test_score = accuracy_score(y_test, y_test_pred)
test_score
```

Out[77]:

```
0.7810676843444313
```

In [78]:

```python
y_train_pred = logreg.predict(x_train)
y_train_pred
```

Out[78]:

```
array([3, 3, 3, ..., 3, 3, 3])
```

In [79]:

```python
# Compare y_train and y_train_pred

train_score = accuracy_score(y_train, y_train_pred)
train_score
```

Out[79]:

```
0.7842165292156519
```

In [82]:

```python
# Predict for accident severity using random numbers

logreg.predict([[4, 3, 40, 9, 6, 9]])
```

Out[82]:

```
array([3])
```