In [1]:

```
# Import required libraries
import pandas as pd
import numpy as np
```

In [2]:

```
# Load the csv into dataframe
accidents = pd.read_csv('ACCIDENT.csv')

# Remove duplicate records if any
accidents.drop_duplicates(inplace=True)

# Replace blank space with NaN value
accidents = accidents.replace(r'^\s*$', np.nan, regex=True)
```

In [3]:

```
# Information regarding the columns
accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54258 entries, 0 to 54257
Data columns (total 28 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   ACCIDENT_NO           54258 non-null  object
 1   ACCIDENTDATE          54258 non-null  object
 2   ACCIDENTTIME          54258 non-null  object
 3   ACCIDENT_TYPE         54258 non-null  int64 
 4   Accident Type Desc    54258 non-null  object
 5   DAY_OF_WEEK           54258 non-null  int64 
 6   Day Week Description  54258 non-null  object
 7   DCA_CODE              54258 non-null  int64 
 8   DCA Description       54258 non-null  object
 9   DIRECTORY             49963 non-null  object
 10  EDITION               49963 non-null  object
 11  PAGE                  49963 non-null  object
 12  GRID_REFERENCE_X      49963 non-null  object
 13  GRID_REFERENCE_Y      49963 non-null  object
 14  LIGHT_CONDITION       54258 non-null  int64 
 15  Light Condition Desc  54258 non-null  object
 16  NODE_ID               54258 non-null  int64 
 17  NO_OF_VEHICLES        54258 non-null  int64 
 18  NO_PERSONS            54258 non-null  int64 
 19  NO_PERSONS_INJ_2      54258 non-null  int64 
 20  NO_PERSONS_INJ_3      54258 non-null  int64 
 21  NO_PERSONS_KILLED     54258 non-null  int64 
 22  NO_PERSONS_NOT_INJ    54258 non-null  int64 
 23  POLICE_ATTEND         54258 non-null  int64 
 24  ROAD_GEOMETRY         54258 non-null  int64 
 25  Road Geometry Desc    54258 non-null  object
 26  SEVERITY              54258 non-null  int64 
 27  SPEED_ZONE            54258 non-null  int64 
dtypes: int64(15), object(13)
memory usage: 12.0+ MB
```

The above result shows that there are `28` columns in total out of which `15` are integer based and `13` are string/object based columns. There are `54,258` rows in this dataframe. It can be seen that there are few nulls in columns `DIRECTORY`, `EDITION`, `PAGE`, `GRID_REFERENCE_X` and `GRID_REFERENCE_Y`.

In [4]:

```
# Checkout the stats for the integer/float based columns
accidents.describe().transpose()
```

Out[4]:

	count	mean	std	min	25%	50%	
ACCIDENT_TYPE	54258.0	2.264053	2.024974	1.0	1.0	1.0	
DAY_OF_WEEK	54258.0	3.897803	2.004751	0.0	2.0	4.0	
DCA_CODE	54258.0	139.743466	26.510306	100.0	120.0	130.0	1
LIGHT_CONDITION	54258.0	1.890818	1.621712	1.0	1.0	1.0	
NODE_ID	54258.0	202216.874894	122815.544310	-10.0	46502.0	277362.5	3022
NO_OF_VEHICLES	54258.0	1.803752	0.748534	1.0	1.0	2.0	
NO_PERSONS	54258.0	2.385436	1.546309	1.0	2.0	2.0	
NO_PERSONS_INJ_2	54258.0	0.329647	0.587104	0.0	0.0	0.0	
NO_PERSONS_INJ_3	54258.0	0.913543	0.763322	0.0	0.0	1.0	
NO_PERSONS_KILLED	54258.0	0.019002	0.149176	0.0	0.0	0.0	
NO_PERSONS_NOT_INJ	54258.0	1.122784	1.354557	0.0	0.0	1.0	
POLICE_ATTEND	54258.0	1.290280	0.655738	1.0	1.0	1.0	
ROAD_GEOMETRY	54258.0	3.406926	1.764163	1.0	2.0	5.0	
SEVERITY	54258.0	2.681909	0.502075	1.0	2.0	3.0	
SPEED_ZONE	54258.0	130.215858	231.491585	30.0	60.0	60.0	

From the above results, we can see that the minimum values `0` and `-10` for `DAY_OF_WEEK` and `NODE_ID` respectively are not expected. These values need to be fixed or ignored for the analysis part (shall decide later). All other column details seem fine for now.

In [5]:

```
# Checkout the stats for the string/object based columns
accidents.describe(include='object').transpose()
```

Out[5]:

	count	unique	top	freq
ACCIDENT_NO	54258	54258	T20150009952	1
ACCIDENTDATE	54258	1471	17/04/2015	76
ACCIDENTTIME	54258	1400	16.00.00	846
Accident Type Desc	54258	9	Collision with vehicle	34160
Day Week Description	54258	7	Friday	8527
DCA Description	54258	80	REAR END(VEHICLES IN SAME LANE)	9783
DIRECTORY	49963	3	MEL	37557
EDITION	49963	4	40	35525
PAGE	49963	584	29	823
GRID_REFERENCE_X	49963	21	H	5176
GRID_REFERENCE_Y	49963	14	9	4685
Light Condition Desc	54258	7	Day	34633
Road Geometry Desc	54258	9	Not at intersection	28276

From the above results, we can deduct information for various questions such as on which date most accidents occurred, at what time most accidents occurred, the most frequent type of accident, on which week-day most accidents took place, most frequent definition of classifying accidents, most frequent level of brightness at the time of accident and most frequent layout of the road when accident took place.

In [6]:

```
# Display first few records
accidents.head()
```

Out[6]:

	ACCIDENT_NO	ACCIDENTDATE	ACCIDENTTIME	ACCIDENT_TYPE	Accident Type Desc	DAY_OF_WEE
0	T20060000265	20/01/2006	22.10.00	4	Collision with a fixed object	
1	T20070034197	18/12/2006	19.30.00	1	Collision with vehicle	
2	T20070035112	14/12/2006	15.55.00	2	Struck Pedestrian	
3	T20070036807	20/05/2006	10.00.00	1	Collision with vehicle	
4	T20070038568	12/11/2006	15.00.00	7	Fall from or in moving vehicle	

5 rows × 28 columns

In [7]:

```
# Display columns
accidents.columns
```

Out[7]:

```
Index(['ACCIDENT_NO', 'ACCIDENTDATE', 'ACCIDENTTIME', 'ACCIDENT_TYPE',
       'Accident Type Desc', 'DAY_OF_WEEK', 'Day Week Description', 'DCA_COD
E',
       'DCA Description', 'DIRECTORY', 'EDITION', 'PAGE', 'GRID_REFERENCE_
X',
       'GRID_REFERENCE_Y', 'LIGHT_CONDITION', 'Light Condition Desc',
       'NODE_ID', 'NO_OF_VEHICLES', 'NO_PERSONS', 'NO_PERSONS_INJ_2',
       'NO_PERSONS_INJ_3', 'NO_PERSONS_KILLED', 'NO_PERSONS_NOT_INJ',
       'POLICE_ATTEND', 'ROAD_GEOMETRY', 'Road Geometry Desc', 'SEVERITY',
       'SPEED_ZONE'],
      dtype='object')
```

For ease, I checked on excel and tableau that the format for datetime and enum for `Accident Type Desc`, `DCA Description` and `Road Geometry Desc` are fine. However, the `ACCIDENT_NO` and `ACCIDENTDATE` aren't in sync. I've checked the `NODE.csv` as well and most accident numbers match except the ones with wrong/bad `NODE_ID`.

For example: Consider record with index `1`. As per the accident number, the year should have been 2007 but it is 2006. Hence, there's a need to fix such issues. Also there are certain columns such as `ACCIDENT_TYPE`, `DAY_OF_WEEK`, `DCA_CODE`, `DIRECTORY`, `EDITION`, `PAGE`, `GRID_REFERENCE_X`, `GRID_REFERENCE_Y`, `LIGHT_CONDITION` and `ROAD_GEOMETRY` that aren't required as they have nulls/irrelevant information.

In [8]:

```
# Choosing a subset of required columns
accidents = accidents[['ACCIDENT_NO', 'ACCIDENTDATE', 'ACCIDENTTIME', 'Accident Type Desc',
                       'Day Week Description', 'DCA Description', 'Light Condition Desc',
       'NODE_ID', 'NO_OF_VEHICLES', 'NO_PERSONS', 'NO_PERSONS_INJ_2',
       'NO_PERSONS_INJ_3', 'NO_PERSONS_KILLED', 'NO_PERSONS_NOT_INJ',
       'POLICE_ATTEND', 'Road Geometry Desc', 'SEVERITY', 'SPEED_ZONE']]

# Split the ACCIDENTDATE column into 2 parts with delimiter as '/' to get day, month and ye
accidentDate = accidents['ACCIDENTDATE'].str.split("/", n = 2, expand = True)

# Assigning day, month and year values we got from the split to new day, month and year col
accidents['day'] = accidentDate[0]
accidents['month'] = accidentDate[1]
accidents['year'] = accidents['ACCIDENT_NO'].str.extract('^T(\d{4})')

# Concat day, month and year with '/' to give it a date format
accidents['ACCIDENTDATE'] = accidents['day'] + '/' + accidents['month'] + '/' + accidents['

# Replace '.' with ':' to make it into time format
accidents['ACCIDENTTIME'] = accidents['ACCIDENTTIME'].str.replace('.', ":")

# Concat accident date and time and convert it into pandas datetime format
accidents['ACCIDENT_DATETIME'] = pd.to_datetime(accidents['ACCIDENTDATE'] + ' ' + accidents

# Get the actual weekday after the change in accident date according to accident number
accidents['WEEKDAY'] = accidents['ACCIDENT_DATETIME'].dt.day_name()

# Change the severity codes into their respective description
accidents['SEVERITY'] = accidents['SEVERITY'].astype(str).replace(str(1), "Fatal Accident")
accidents['SEVERITY'] = accidents['SEVERITY'].astype(str).replace(str(2), "Serious Injury A
accidents['SEVERITY'] = accidents['SEVERITY'].astype(str).replace(str(3), "Other Injury Acc
accidents['SEVERITY'] = accidents['SEVERITY'].astype(str).replace(str(4), "Non Injury Accid


# Lastly, picking required columns only
accidents = accidents[['ACCIDENT_NO', 'ACCIDENT_DATETIME', 'Accident Type Desc',
                       'WEEKDAY', 'DCA Description', 'Light Condition Desc',
       'NODE_ID', 'NO_OF_VEHICLES', 'NO_PERSONS', 'NO_PERSONS_INJ_2',
       'NO_PERSONS_INJ_3', 'NO_PERSONS_KILLED', 'NO_PERSONS_NOT_INJ',
       'POLICE_ATTEND', 'Road Geometry Desc', 'SEVERITY', 'SPEED_ZONE']]

# Check for nulls
accidents.isnull().sum()
```

Out[8]:

```
ACCIDENT_NO             0
ACCIDENT_DATETIME       0
Accident Type Desc      0
WEEKDAY                 0
DCA Description         0
Light Condition Desc    0
NODE_ID                 0
NO_OF_VEHICLES          0
NO_PERSONS              0
NO_PERSONS_INJ_2        0
NO_PERSONS_INJ_3        0
NO_PERSONS_KILLED       0
NO_PERSONS_NOT_INJ      0
```

```
POLICE_ATTEND           0
Road Geometry Desc      0
SEVERITY                0
SPEED_ZONE              0
dtype: int64
```

In [9]:

```
# Display first few records
accidents.head()
```

Out[9]:

	ACCIDENT_NO	ACCIDENT_DATETIME	Accident Type Desc	WEEKDAY	DCA Description	Light Condition Desc	N
0	T20060000265	2006-01-20 22:10:00	Collision with a fixed object	Friday	LEFT OFF CARRIAGEWAY INTO OBJECT/PARKED VEHICL...	Dark No street lights	
1	T20070034197	2007-12-18 19:30:00	Collision with vehicle	Tuesday	RIGHT THROUGH	Day	
2	T20070035112	2007-12-14 15:55:00	Struck Pedestrian	Friday	ANY MANOEUVRE INVOLVING PED NOT INCLUDED IN DC...	Day	
3	T20070036807	2007-05-20 10:00:00	Collision with vehicle	Sunday	LEAVING PARKING	Day	
4	T20070038568	2007-12-11 15:00:00	Fall from or in moving vehicle	Tuesday	OUT OF CONTROL ON CARRIAGEWAY (ON STRAIGHT) ...	Day	

Now that I've worked on the accidents main file, let's explore `NODE.csv` file.

In [10]:

```
# Load the csv into dataframe
node = pd.read_csv('NODE.csv')

# Drop duplicate records if any
node.drop_duplicates(inplace=True)

# Replace the blank spaces with NaN values
node = node.replace(r'^\s*$', np.nan, regex=True)
```

In [11]:

```
# Information regarding the columns
node.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 166602 entries, 0 to 166601
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   ACCIDENT_NO     166602 non-null  object 
 1   NODE_ID         166602 non-null  int64  
 2   NODE_TYPE       166582 non-null  object 
 3   AMG_X           166602 non-null  float64
 4   AMG_Y           166602 non-null  float64
 5   LGA_NAME        166567 non-null  object 
 6   Lga Name All    166602 non-null  object 
 7   Region Name     166561 non-null  object 
 8   Deg Urban Name  166602 non-null  object 
 9   Lat             166602 non-null  float64
 10  Long            166602 non-null  float64
 11  Postcode No     166602 non-null  int64  
dtypes: float64(4), int64(2), object(6)
memory usage: 16.5+ MB
```

From the above results, we can see that there are `166,602` records which is much more than the `ACCIDENT.csv` records. This implies that either there are many other nodes where no accidents occurred.

There are a total of `12` columns out of which `4` are floating-point based, `2` are integer based and `6` are string/object. There are few null values in columns `NODE_TYPE`, `LGA_NAME` and `Region Name`.

In [12]:

```
# Checkout the stats for the integer based columns
node.describe().transpose()
```

Out[12]:

	count	mean	std	min	25%	50%
NODE_ID	166602.0	1.618483e+05	111057.582820	4.000000e+00	4.292600e+04	2.107955e+05
AMG_X	166602.0	2.498054e+06	72224.724328	2.129297e+06	2.488541e+06	2.501675e+06
AMG_Y	166602.0	2.420624e+06	63064.599544	2.273614e+06	2.394272e+06	2.409205e+06
Lat	166602.0	-3.771247e+01	0.571074	-3.903232e+01	-3.795167e+01	-3.781790e+01
Long	166602.0	1.449791e+02	0.813129	1.409662e+02	1.448699e+02	1.450190e+02
Postcode No	166602.0	3.310207e+03	302.772622	3.000000e+03	3.074000e+03	3.175000e+03

In [13]:

```
# Checkout the stats for the string/object based columns
node.describe(include= 'object').transpose()
```

Out[13]:

	count	unique	top	freq
ACCIDENT_NO	166602	163913	T20070022419	3
NODE_TYPE	166582	3	N	86272
LGA_NAME	166567	87	MELBOURNE	10475
Lga Name All	166602	231	MELBOURNE	9216
Region Name	166561	7	METROPOLITAN SOUTH EAST REGION	61273
Deg Urban Name	166602	7	MELB_URBAN	105663

From the results of the above two cells, we can see that there are multiple `ACCIDENT_NO` , most frequent `NODE_TYPE` is `Non-Intersection` with `86272` occurrences and pretty much all the nodes are in and around `Melbourne` (specifically, VIC region) based on the `Local Government Area (LGA)` name.

In [14]:

```
# To see which accident numbers have how many records
node['ACCIDENT_NO'].value_counts()
```

Out[14]:

```
T20070022419    3
T20090029507    3
T20130026885    3
T20140020930    3
T20110023705    3
               ..
T20060034936    1
T20110006499    1
T20160016134    1
T20080038215    1
T20160026228    1
Name: ACCIDENT_NO, Length: 163913, dtype: int64
```

In [15]:

```
# Multiple entries with same accident no and node id -- need to fix this otherwise it'll le
node[node['ACCIDENT_NO'] == 'T20170000121']
```

Out[15]:

	ACCIDENT_NO	NODE_ID	NODE_TYPE	AMG_X	AMG_Y	LGA_NAME	Lga Nai
153847	T20170000121	24437	I	2538376.485	2381942.623	CARDINIA	CARDIN
153848	T20170000121	24437	I	2538376.485	2381942.623	CARDINIA	CARDIN
153849	T20170000121	24437	I	2538376.485	2381942.623	CARDINIA	CARDIN

It seems that the `Deg Urban Name` is the only field that is different among these rows. Since this field doesn't serves my analysis, I'm going to remove this column and drop the duplicates within the dataframe.

In [16]:

```
# Accident numbers with bad/wrong node ids
suspiciousAccidentNo = accidents[accidents['NODE_ID'] <= 0]['ACCIDENT_NO'].tolist()

# To see if the suspicious accident numbers are present in the node file (the full list of
node[node['ACCIDENT_NO'].isin(suspiciousAccidentNo)]
```

Out[16]:

	ACCIDENT_NO	NODE_ID	NODE_TYPE	AMG_X	AMG_Y	LGA_NAME	Lga Name All	Region Name	Deg Urban Name

As mentioned earlier about the wrong `NODE_ID`, I looked up the suspicious `ACCIDENT_NO` in the node dataframe and found no records. This implies that the suspicious accident number have unknown nodes which again doesn't serve our analytics purpose. So, I'll discard them.

In the next cell, I'll replace the node type with its description and whereever there are null/blank values, I'll replace them with `Unknown` as there is no other way to determine.

In [17]:

```
# Change the node type codes into their respective description
node['NODE_TYPE'] = node['NODE_TYPE'].str.replace(str('I'), "Intersection")
node['NODE_TYPE'] = node['NODE_TYPE'].str.replace(str('N'), "Non-Intersection")
node['NODE_TYPE'] = node['NODE_TYPE'].str.replace(str('O'), "Off Road")
node['NODE_TYPE'] = node['NODE_TYPE'].fillna('Unknown')

# Display first few records
node.head()
```

Out[17]:

	ACCIDENT_NO	NODE_ID	NODE_TYPE	AMG_X	AMG_Y	LGA_NAME	
0	T20060000010	43078	Intersection	2519154.655	2390265.154	DANDENONG	
1	T20060000018	29720	Non-Intersection	2524272.743	2389996.817	CASEY	
2	T20060000022	203074	Non-Intersection	2487321.693	2345019.925	MORNINGTON PENINSULA	
3	T20060000023	55462	Intersection	2512734.120	2390214.959	DANDENONG	KINGSTON
4	T20060000026	202988	Non-Intersection	2488777.662	2347612.069	MORNINGTON PENINSULA	

Also, as the columns `AMG_X`, `AMG_Y`, `Region Name`, `LGA_NAME` (I looked up in excel as well and found that the nulls aren't reflecting in there as they are treated as empty space. However, `Lga Name All` has non-null values throughout) and `Deg Urban Name` serve no purpose for the analytics, I'll discard them.

In [18]:

```
# Choosing the subset of required columns only
node = node[['ACCIDENT_NO', 'NODE_ID', 'NODE_TYPE', 'Lga Name All', 'Lat', 'Long', 'Postcod

# Drop the duplicates
node.drop_duplicates(inplace=True)

# Check for any null values in the columns
node.isnull().sum()
```

Out[18]:

```
ACCIDENT_NO     0
NODE_ID         0
NODE_TYPE       0
Lga Name All    0
Lat             0
Long            0
Postcode No     0
dtype: int64
```

In [19]:

```
# Inner join the accidents and node dataframes on accident number and node id
accidentNodes = pd.merge(accidents, node, on=['ACCIDENT_NO', 'NODE_ID'], how='inner')

# Display first few records
accidentNodes.head()
```

Out[19]:

	ACCIDENT_NO	ACCIDENT_DATETIME	Accident Type Desc	WEEKDAY	DCA Description	Light Condition Desc	NO
0	T20070034197	2007-12-18 19:30:00	Collision with vehicle	Tuesday	RIGHT THROUGH	Day	
1	T20070035112	2007-12-14 15:55:00	Struck Pedestrian	Friday	ANY MANOEUVRE INVOLVING PED NOT INCLUDED IN DC...	Day	2
2	T20070036807	2007-05-20 10:00:00	Collision with vehicle	Sunday	LEAVING PARKING	Day	
3	T20070038568	2007-12-11 15:00:00	Fall from or in moving vehicle	Tuesday	OUT OF CONTROL ON CARRIAGEWAY (ON STRAIGHT) ...	Day	2
4	T20070040775	2007-11-13 08:45:00	Collision with vehicle	Tuesday	OTHER SAME DIRECTION-MANOUEVRES NOT INCLUDED I...	Day	2

5 rows × 22 columns

In [20]:

```
# To see if the duplicate records are dropped
accidentNodes[accidentNodes['ACCIDENT_NO'] == 'T20170000121']
```

Out[20]:

	ACCIDENT_NO	ACCIDENT_DATETIME	Accident Type Desc	WEEKDAY	DCA Description	Light Condition Desc	NOI
42728	T20170000121	2017-12-17 22:50:00	Collision with vehicle	Sunday	RIGHT THROUGH	Dark Street lights on	

1 rows × 22 columns

In [21]:

```
# To see if the records with wrong nodes are not present due to inner join
accidentNodes[accidentNodes['ACCIDENT_NO'] == 'T20060000265']
```

Out[21]:

ACCIDENT_NO	ACCIDENT_DATETIME	Accident Type Desc	WEEKDAY	DCA Description	Light Condition Desc	NODE_ID

0 rows × 22 columns

In [22]:

```
# To check for nulls after join
accidentNodes.isnull().sum()
```

Out[22]:

```
ACCIDENT_NO             0
ACCIDENT_DATETIME       0
Accident Type Desc      0
WEEKDAY                 0
DCA Description         0
Light Condition Desc    0
NODE_ID                 0
NO_OF_VEHICLES          0
NO_PERSONS              0
NO_PERSONS_INJ_2        0
NO_PERSONS_INJ_3        0
NO_PERSONS_KILLED       0
NO_PERSONS_NOT_INJ      0
POLICE_ATTEND           0
Road Geometry Desc      0
SEVERITY                0
SPEED_ZONE              0
NODE_TYPE               0
Lga Name All            0
Lat                     0
Long                    0
Postcode No             0
dtype: int64
```

In [23]:

```
# Information regarding the columns
accidentNodes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54244 entries, 0 to 54243
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ACCIDENT_NO           54244 non-null  object
 1   ACCIDENT_DATETIME     54244 non-null  datetime64[ns]
 2   Accident Type Desc    54244 non-null  object
 3   WEEKDAY               54244 non-null  object
 4   DCA Description       54244 non-null  object
 5   Light Condition Desc  54244 non-null  object
 6   NODE_ID               54244 non-null  int64
 7   NO_OF_VEHICLES        54244 non-null  int64
 8   NO_PERSONS            54244 non-null  int64
 9   NO_PERSONS_INJ_2      54244 non-null  int64
 10  NO_PERSONS_INJ_3      54244 non-null  int64
 11  NO_PERSONS_KILLED     54244 non-null  int64
 12  NO_PERSONS_NOT_INJ    54244 non-null  int64
 13  POLICE_ATTEND         54244 non-null  int64
 14  Road Geometry Desc    54244 non-null  object
 15  SEVERITY              54244 non-null  object
 16  SPEED_ZONE            54244 non-null  int64
 17  NODE_TYPE             54244 non-null  object
 18  Lga Name All          54244 non-null  object
 19  Lat                   54244 non-null  float64
 20  Long                  54244 non-null  float64
 21  Postcode No           54244 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(10), object(9)
memory usage: 9.5+ MB
```

So the merged dataframe has a total of `54,244` records and `22` columns out of which `1` is datetime based, `2` are floating-point based, `9` are integer based and `10` are string/object based columns. Lastly, there are no null values.

Now that I've dealt with the accident and node files, I'll look up accident location file.

In [24]:

```
# Load the csv into dataframe
accidentLocation = pd.read_csv('ACCIDENT_LOCATION.csv')

# Drop duplicate records if any
accidentLocation.drop_duplicates(inplace=True)

# Replace the blank spaces with NaN values
accidentLocation = accidentLocation.replace(r'^\s*$', np.nan, regex=True)
```

In [25]:

```
# To check for nulls
accidentLocation.isnull().sum()
```

Out[25]:

```
ACCIDENT_NO               0
NODE_ID                   0
ROAD_ROUTE_1            876
ROAD_NAME              5252
ROAD_TYPE              6999
ROAD_NAME_INT          1332
ROAD_TYPE_INT          2667
DISTANCE_LOCATION       876
DIRECTION_LOCATION      917
NEAREST_KM_POST      160511
OFF_ROAD_LOCATION    148624
dtype: int64
```

In [26]:

```
# Information regarding the columns
accidentLocation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 164789 entries, 0 to 164788
Data columns (total 11 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   ACCIDENT_NO         164789 non-null  object
 1   NODE_ID             164789 non-null  int64
 2   ROAD_ROUTE_1        163913 non-null  float64
 3   ROAD_NAME           159537 non-null  object
 4   ROAD_TYPE           157790 non-null  object
 5   ROAD_NAME_INT       163457 non-null  object
 6   ROAD_TYPE_INT       162122 non-null  object
 7   DISTANCE_LOCATION   163913 non-null  float64
 8   DIRECTION_LOCATION  163872 non-null  object
 9   NEAREST_KM_POST     4278 non-null    float64
 10  OFF_ROAD_LOCATION   16165 non-null   object
dtypes: float64(3), int64(1), object(7)
memory usage: 15.1+ MB
```

There are a total of `164,789` records and `11` columns out of which `3` are floating-point based, `1` is integer based and `7` are object based columns. It seems there a lot of null values in this dataframe in most columns. Considering this, as most of the data from this dataframe is not helpful, I'll discard all columns expect for `ACCIDENT_NO`, `NODE_ID` and `ROAD_ROUTE_1` (this needs fixing).

In [27]:

```
# Checkout the stats for the integer/float based columns
accidentLocation.describe().transpose()
```

Out[27]:

	count	mean	std	min	25%	50%	75
NODE_ID	164789.0	161455.360418	111355.684234	-10.0	42723.0	210679.0	261147
ROAD_ROUTE_1	163913.0	6272.846595	3110.580864	-1.0	2880.0	5770.0	9999
DISTANCE_LOCATION	163913.0	130.126884	391.138150	-1.0	0.0	13.0	80
NEAREST_KM_POST	4278.0	0.000000	0.000000	0.0	0.0	0.0	0

In [28]:

```
# Checkout the stats for the string/object based columns
accidentLocation.describe(include='object').transpose()
```

Out[28]:

	count	unique	top	freq
ACCIDENT_NO	164789	164789	T20150002609	1
ROAD_NAME	159537	12429	PRINCES	4103
ROAD_TYPE	157790	69	ROAD	80328
ROAD_NAME_INT	163457	19721	VICTORIA	1022
ROAD_TYPE_INT	162122	102	ROAD	60705
DIRECTION_LOCATION	163872	9	W	37220
OFF_ROAD_LOCATION	16165	4794	1;;;;	3435

Following the `DATA_DICTIONARY.csv` , I'll update the `ROAD_ROUTE_1` column with the respective descriptions into new column `ROAD_ROUTE` and discard old column.

In [29]:

```
# Fill null values in road route with 0
accidentLocation['ROAD_ROUTE_1'] = accidentLocation['ROAD_ROUTE_1'].fillna(0)

# To see if null values have been replaced
accidentLocation[accidentLocation['ROAD_ROUTE_1'].isna()]
```

Out[29]:

ACCIDENT_NO	NODE_ID	ROAD_ROUTE_1	ROAD_NAME	ROAD_TYPE	ROAD_NAME_INT	RO

In [30]:

```
# To see which and how many road routes out of the assigned numbers in the data dictionary
accidentLocation[(accidentLocation['ROAD_ROUTE_1'] < 2000) |
                 ((accidentLocation['ROAD_ROUTE_1'] >= 6000) & (accidentLocation['ROAD_ROUT
                ((accidentLocation['ROAD_ROUTE_1'] >= 8000) & (accidentLocation['ROAD_ROUTE
                (accidentLocation['ROAD_ROUTE_1'] > 9999)]['ROAD_ROUTE_1'].value_counts()
```

Out[30]:

```
 0.0       5152
-1.0        273
 8012.0      22
 8010.0      16
 8053.0      14
 8011.0       7
 8026.0       5
 8025.0       4
 8005.0       4
 8040.0       3
 8001.0       3
 8024.0       2
 8022.0       2
 8014.0       2
 8009.0       2
 8042.0       1
 8044.0       1
 8002.0       1
 8004.0       1
 8013.0       1
 8041.0       1
 8037.0       1
 8015.0       1
 8047.0       1
 8021.0       1
 8027.0       1
 8016.0       1
Name: ROAD_ROUTE_1, dtype: int64
```

In [31]:

```
# Assign the description based on road route code in data dictionary file

# For unknown
accidentLocation.loc[(accidentLocation['ROAD_ROUTE_1'] < 2000) |
                ((accidentLocation['ROAD_ROUTE_1'] >= 6000) & (accidentLocation['ROAD_ROUT
               ((accidentLocation['ROAD_ROUTE_1'] >= 8000) & (accidentLocation['ROAD_ROUTE
               (accidentLocation['ROAD_ROUTE_1'] > 9999),
                     'ROAD_ROUTE'] = 'Unknown'

# For Freeways or Highways
accidentLocation.loc[((accidentLocation['ROAD_ROUTE_1'] >= 2000) & (accidentLocation['ROAD_
                     'ROAD_ROUTE'] = 'Freeways or Highways'

# For Forest Roads
accidentLocation.loc[((accidentLocation['ROAD_ROUTE_1'] >= 3000) & (accidentLocation['ROAD_
                     'ROAD_ROUTE'] = 'Forest Roads'

# For Tourist Roads
accidentLocation.loc[((accidentLocation['ROAD_ROUTE_1'] >= 4000) & (accidentLocation['ROAD_
                     'ROAD_ROUTE'] = 'Tourist Roads'

# For Main Roads
accidentLocation.loc[((accidentLocation['ROAD_ROUTE_1'] >= 5000) & (accidentLocation['ROAD_
                     'ROAD_ROUTE'] = 'Main Roads'

# For Ramps
accidentLocation.loc[((accidentLocation['ROAD_ROUTE_1'] >= 7000) & (accidentLocation['ROAD_
                     'ROAD_ROUTE'] = 'Ramps'

# For Unclassified Roads
accidentLocation.loc[(accidentLocation['ROAD_ROUTE_1'] == 9999),
                     'ROAD_ROUTE'] = 'Unclassified Roads'
```

In [32]:

```
# Display columns
accidentLocation.columns
```

Out[32]:

```
Index(['ACCIDENT_NO', 'NODE_ID', 'ROAD_ROUTE_1', 'ROAD_NAME', 'ROAD_TYPE',
       'ROAD_NAME_INT', 'ROAD_TYPE_INT', 'DISTANCE_LOCATION',
       'DIRECTION_LOCATION', 'NEAREST_KM_POST', 'OFF_ROAD_LOCATION',
       'ROAD_ROUTE'],
      dtype='object')
```

After fixing the `ROAD_ROUTE` column, I'll discard all other columns except for `ACCIDENT_NO` and `NODE_ID` as most of the values in other columns are nulls which can't be used for analysis later.

In [33]:

```
# Choosing subset of required columns
accidentLocation = accidentLocation[['ACCIDENT_NO', 'NODE_ID', 'ROAD_ROUTE']]

# Filter the suspicious records with bad/wrong node ids
suspiciousAccidentLocationNo = accidentLocation[accidentLocation['NODE_ID'] <= 0]['ACCIDENT

# To check if these suspicious accident numbers exist in node file which is the mail file w
node[node['ACCIDENT_NO'].isin(suspiciousAccidentLocationNo)]
```

Out[33]:

ACCIDENT_NO	NODE_ID	NODE_TYPE	Lga Name All	Lat	Long	Postcode No

Upon merging the dataframes, these records will not a part of final dataframe as node id wouldn't match which was earlier matched with the `NODE_ID` from `node` dataframe.

In [34]:

```
# As there are no suspicious records, we can go ahead and merge accidentNodes and accidentL
finalDf = pd.merge(accidentNodes, accidentLocation, on=['ACCIDENT_NO', 'NODE_ID'], how='lef

# Display first few records
finalDf.head()
```

Out[34]:

	ACCIDENT_NO	ACCIDENT_DATETIME	Accident Type Desc	WEEKDAY	DCA Description	Light Condition Desc	NO
0	T20070034197	2007-12-18 19:30:00	Collision with vehicle	Tuesday	RIGHT THROUGH	Day	
1	T20070035112	2007-12-14 15:55:00	Struck Pedestrian	Friday	ANY MANOEUVRE INVOLVING PED NOT INCLUDED IN DC...	Day	2
2	T20070036807	2007-05-20 10:00:00	Collision with vehicle	Sunday	LEAVING PARKING	Day	
3	T20070038568	2007-12-11 15:00:00	Fall from or in moving vehicle	Tuesday	OUT OF CONTROL ON CARRIAGEWAY (ON STRAIGHT) ...	Day	2
4	T20070040775	2007-11-13 08:45:00	Collision with vehicle	Tuesday	OTHER SAME DIRECTION-MANOUEVRES NOT INCLUDED I...	Day	2

5 rows × 23 columns

In [35]:

```
# To see if any nulls exists in dataframe
finalDf.isnull().values.any()
```

Out[35]:

```
False
```

In [36]:

```
# Information regarding the columns
finalDf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54244 entries, 0 to 54243
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   ACCIDENT_NO           54244 non-null  object        
 1   ACCIDENT_DATETIME     54244 non-null  datetime64[ns]
 2   Accident Type Desc    54244 non-null  object        
 3   WEEKDAY               54244 non-null  object        
 4   DCA Description       54244 non-null  object        
 5   Light Condition Desc  54244 non-null  object        
 6   NODE_ID               54244 non-null  int64         
 7   NO_OF_VEHICLES        54244 non-null  int64         
 8   NO_PERSONS            54244 non-null  int64         
 9   NO_PERSONS_INJ_2      54244 non-null  int64         
 10  NO_PERSONS_INJ_3      54244 non-null  int64         
 11  NO_PERSONS_KILLED     54244 non-null  int64         
 12  NO_PERSONS_NOT_INJ    54244 non-null  int64         
 13  POLICE_ATTEND         54244 non-null  int64         
 14  Road Geometry Desc    54244 non-null  object        
 15  SEVERITY              54244 non-null  object        
 16  SPEED_ZONE            54244 non-null  int64         
 17  NODE_TYPE             54244 non-null  object        
 18  Lga Name All          54244 non-null  object        
 19  Lat                   54244 non-null  float64       
 20  Long                  54244 non-null  float64       
 21  Postcode No           54244 non-null  int64         
 22  ROAD_ROUTE            54244 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(10), object(10)
memory usage: 9.9+ MB
```

In [37]:

```
# Checkout the stats for the integer/float based columns
finalDf.describe().transpose()
```

Out[37]:

	count	mean	std	min	25%	
NODE_ID	54244.0	202263.292604	122790.554829	4.00000	46536.250000	2773
NO_OF_VEHICLES	54244.0	1.803849	0.748490	1.00000	1.000000	
NO_PERSONS	54244.0	2.385536	1.546353	1.00000	2.000000	
NO_PERSONS_INJ_2	54244.0	0.329714	0.587151	0.00000	0.000000	
NO_PERSONS_INJ_3	54244.0	0.913447	0.763290	0.00000	0.000000	
NO_PERSONS_KILLED	54244.0	0.019007	0.149195	0.00000	0.000000	
NO_PERSONS_NOT_INJ	54244.0	1.122908	1.354606	0.00000	0.000000	
POLICE_ATTEND	54244.0	1.290244	0.655771	1.00000	1.000000	
SPEED_ZONE	54244.0	130.134042	231.361452	30.00000	60.000000	
Lat	54244.0	-37.707053	0.572090	-39.02399	-37.949852	-
Long	54244.0	144.972959	0.815913	140.96648	144.850030	1
Postcode No	54244.0	3312.631074	302.405969	3000.00000	3074.000000	31

In [38]:

```
# Checkout the stats for the string/object based columns
finalDf.describe(include='object').transpose()
```

Out[38]:

	count	unique	top	freq
ACCIDENT_NO	54244	54244	T20150009952	1
Accident Type Desc	54244	9	Collision with vehicle	34156
WEEKDAY	54244	7	Friday	8305
DCA Description	54244	80	REAR END(VEHICLES IN SAME LANE)	9781
Light Condition Desc	54244	7	Day	34626
Road Geometry Desc	54244	9	Not at intersection	28266
SEVERITY	54244	4	Other Injury Accident	37934
NODE_TYPE	54244	4	Non-Intersection	29328
Lga Name All	54244	210	MELBOURNE	2609
ROAD_ROUTE	54244	7	Unclassified Roads	18229

So, the final dataframe has no null values and all the column values seem fine. I'll now transfer the content into a excel file to visualise further on tableau.

In [39]:

```
finalDf.to_excel('Final Dataset.xlsx', index=False)
```