IMPORT LIBRARIES

In [1]:
```
import sqlite3
import pandas as pd
import numpy as np
!pip install matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

!pip install warnings
import warnings
warnings.filterwarnings('ignore')
!pip install mlxtend
import mlxtend
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

```
Requirement already satisfied: matplotlib in c:\users\kemi\appdata\local\progra
ms\python\python311\lib\site-packages (3.7.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\kemi\appdata\local\
programs\python\python311\lib\site-packages (from matplotlib) (1.0.7)
Requirement already satisfied: cycler>=0.10 in c:\users\kemi\appdata\local\prog
rams\python\python311\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from matplotlib) (4.38.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\users\kemi\appdata\local\progr
ams\python\python311\lib\site-packages (from matplotlib) (1.24.2)
Requirement already satisfied: packaging>=20.0 in c:\users\kemi\appdata\local\p
rograms\python\python311\lib\site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\kemi\appdata\local\pro
grams\python\python311\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\kemi\appdata\local\
programs\python\python311\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kemi\appdata\lo
cal\programs\python\python311\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\kemi\appdata\local\program
s\python\python311\lib\site-packages (from python-dateutil>=2.7->matplotlib)
(1.16.0)

[notice] A new release of pip available: 22.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
ERROR: Could not find a version that satisfies the requirement warnings (from v
ersions: none)
ERROR: No matching distribution found for warnings

[notice] A new release of pip available: 22.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip

Collecting mlxtend
  Downloading mlxtend-0.23.1-py3-none-any.whl (1.4 MB)
     ---------------------------------------- 1.4/1.4 MB 4.8 MB/s eta 0:00:00
Requirement already satisfied: scipy>=1.2.1 in c:\users\kemi\appdata\local\prog
rams\python\python311\lib\site-packages (from mlxtend) (1.10.1)
Requirement already satisfied: numpy>=1.16.2 in c:\users\kemi\appdata\local\pro
grams\python\python311\lib\site-packages (from mlxtend) (1.24.2)
Requirement already satisfied: pandas>=0.24.2 in c:\users\kemi\appdata\local\pr
ograms\python\python311\lib\site-packages (from mlxtend) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\kemi\appdata\loc
al\programs\python\python311\lib\site-packages (from mlxtend) (1.2.2)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\kemi\appdata\loca
l\programs\python\python311\lib\site-packages (from mlxtend) (3.7.0)
Requirement already satisfied: joblib>=0.13.2 in c:\users\kemi\appdata\local\pr
ograms\python\python311\lib\site-packages (from mlxtend) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\kemi\appdata\local\
programs\python\python311\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(1.0.7)
Requirement already satisfied: cycler>=0.10 in c:\users\kemi\appdata\local\prog
```

In [2]:
```
con = sqlite3.connect("accident_data_v1.0.0_2023.db")
```
In [3]:
```
cur = con.cursor()
```

Checking the tables under consideration

In [5]:
```
cur.execute("SELECT name FROM sqlite_master WHERE type='table';")
print(cur.fetchall())
```

```
[('accident',), ('casualty',), ('vehicle',), ('lsoa',)]
```

Accident Data

In [158]:
```
# Displaying the Accident Data

ACC = pd.read_sql("""
                SELECT
                    *
                FROM accident
                """, con)
ACC
```

Out[15

	accic	accic	accic	locat	locat	long	latitu	polic	accic	num	...	pede	light	weat	road	spec	carri	urba	did_	trun	ls
0	2017	2017	010C	5329	1963	-0.08(	51.6!	1	1	2	...	0	4	1	1	0	0	1	1	2	E(
1	2017	2017	010C	5267	1819	-0.17:	51.5:	1	3	2	...	0	4	1	2	0	0	1	1	2	E(
2	2017	2017	010C	5352	1812	-0.05:	51.5'	1	3	3	...	0	4	1	1	0	0	1	1	2	E(
3	2017	2017	010C	5343	1935	-0.06(	51.6:	1	3	2	...	4	4	2	2	0	0	1	1	2	E(
4	2017	2017	010C	5336	1878	-0.07:	51.5:	1	2	1	...	5	4	1	2	0	0	1	1	2	E(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4613	202C	2020	991C	3430	7316	-2.92(	56.4:	99	2	2	...	0	1	1	1	0	0	1	1	-1	-'
4613	202C	2020	991C	2579	6588	-4.26:	55.8(	99	3	1	...	0	1	1	1	0	0	1	2	-1	-'
4613	202C	2020	991C	3836	8106	-2.27'	57.1(	99	2	2	...	0	1	1	1	0	0	2	1	-1	-'
4613	202C	2020	991C	2771	6748	-3.96(	55.9!	99	3	2	...	0	1	1	1	0	0	1	2	-1	-'
4613	202C	2020	991C	2404	6819	-4.56'	56.0(	99	3	1	...	0	1	1	1	0	2	1	1	-1	-'

Vehicle Data

In [8]:
```
# Displaying the Vehicle Data

VEH = pd.read_sql("""
```

```
                              SELECT
                                  *
                              FROM vehicle
                              """, con)
    VEH
```

Out[

	vehi	accic	accic	accic	vehi	vehi	towi	vehi	vehi	vehi	...	jouri	sex_	age_	age_	engi	prop	age_	gene	drive	d
0	0	2017	2017	0100	1	9	0	18	1	5	...	6	1	24	5	1997	2	1	-1	-1	-
1	1	2017	2017	0100	2	2	0	18	1	5	...	6	1	19	4	-1	-1	-1	-1	-1	-
2	2	2017	2017	0100	1	9	0	18	5	1	...	6	1	33	6	1797	8	8	-1	9	1
3	3	2017	2017	0100	2	9	0	18	5	1	...	6	1	40	7	2204	2	12	-1	2	1
4	4	2017	2017	0100	1	9	0	18	3	7	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
849(	8490	2020	202(	9910	1	9	0	7	8	2	...	1	1	57	9	1968	2	2	AUD A5	7	1
849(	8490	2020	202(	9910	2	5	0	16	6	2	...	5	1	38	7	1301	1	2	KTM 1290 SUPE	9	2
849(	8490	2020	202(	9910	1	9	0	7	8	2	...	6	2	68	10	1995	2	1	BMW X3	5	1
849(	8490	2020	202(	9910	2	1	0	18	6	2	...	6	1	76	11	-1	-1	-1	-1	9	1
849(	8490	2020	202(	9910	1	9	0	1	8	4	...	6	1	39	7	999	1	2	FORI FOCU	7	1

Casualty Data

In [9]:
```
# Displaying the Casualty Data

CAS = pd.read_sql("""
                              SELECT
                                  *
                              FROM casualty
                              """, con)
CAS
```

Out|

	casu	accid	accid	accid	vehic	casu	casu	sex_c	age_(	age_l	casu	pede	pede	car_p	bus_(	pede	casu	casu	ca
0	0	2017(	2017	0100(	1	1	2	2	18	4	3	0	0	1	0	0	9	1	2
1	1	2017(	2017	0100(	2	2	1	1	19	4	2	0	0	0	0	0	2	-1	-1
2	2	2017(	2017	0100(	2	3	2	1	18	4	1	0	0	0	0	0	2	-1	-1
3	3	2017(	2017	0100(	1	1	2	2	33	6	3	0	0	1	0	0	9	1	5
4	4	2017(	2017	0100(	3	1	1	2	31	6	3	0	0	0	0	0	9	1	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6003	6003	2020	2020	9910	2	1	1	1	11	3	2	0	0	0	0	0	1	1	2
6003	6003	2020	2020	9910	1	1	3	2	63	9	3	10	1	0	0	0	0	1	10
6003	6003	2020	2020	9910	2	1	1	1	38	7	2	0	0	0	0	0	5	2	9
6003	6003	2020	2020	9910	2	1	1	1	76	11	3	0	0	0	0	0	1	1	9
6003	6003	2020	2020	9910	1	1	3	1	48	8	3	9	9	0	0	0	0	1	1

ROAD TRAFFIC ACCIDENTS DATA IN 2020

In [159]:
```
# Accident tables for 2020

ACC2020 = ACC[ACC["accident_year"] == 2020]
VEH2020 = VEH[VEH["accident_year"] == 2020]
CAS2020 = CAS[CAS["accident_year"] == 2020]
```

Accident data 2020

In [160]:
```
# Accident data 2020
ACC2020
```

Out[16

	acci	acci	acci	loca	loca	long	latit	poli	acci	num	...	pede	light	wea	road	spec	carri	urba	did_	trun	ls
3701	2020	2020	0102	5213	1751	-0.254	51.46	1	3	1	...	9	1	9	9	0	0	1	3	2	E(
3701	2020	2020	0102	5293	1762	-0.139	51.47	1	3	1	...	4	1	1	1	0	0	1	1	2	E(
3701	2020	2020	0102	5264	1827	-0.178	51.52	1	3	1	...	0	4	1	2	0	0	1	1	2	E(
3701	2020	2020	0102	5386	1843	-0.001	51.54	1	2	1	...	4	4	1	1	0	0	1	1	2	E(
3701	2020	2020	0102	5293	1812	-0.137	51.51	1	3	1	...	0	4	1	1	0	0	1	1	2	E(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4613	2020	2020	9910	3430	7316	-2.926	56.47	99	2	2	...	0	1	1	1	0	0	1	1	-1	-1
4613	2020	2020	9910	2579	6588	-4.267	55.80	99	3	1	...	0	1	1	1	0	0	1	2	-1	-1
4613	2020	2020	9910	3836	8106	-2.271	57.18	99	2	2	...	0	1	1	1	0	0	2	1	-1	-1
4613	2020	2020	9910	2771	6748	-3.968	55.95	99	3	2	...	0	1	1	1	0	0	1	2	-1	-1
4613	2020	2020	9910	2404	6819	-4.561	56.00	99	3	1	...	0	1	1	1	0	2	1	1	-1	-1

In [15]:
```
ACC2020.count()
```

Out[15]:

```
accident_index                                 91199
accident_year                                  91199
accident_reference                             91199
location_easting_osgr                          91185
location_northing_osgr                         91185
longitude                                      91185
latitude                                       91185
police_force                                   91199
accident_severity                              91199
number_of_vehicles                             91199
number_of_casualties                           91199
date                                           91199
day_of_week                                    91199
time                                           91199
local_authority_district                       91199
local_authority_ons_district                   91199
local_authority_highway                        91199
first_road_class                               91199
first_road_number                              91199
road_type                                      91199
speed_limit                                    91199
junction_detail                                91199
junction_control                               91199
second_road_class                              91199
second_road_number                             91199
pedestrian_crossing_human_control              91199
pedestrian_crossing_physical_facilities        91199
light_conditions                               91199
weather_conditions                             91199
road_surface_conditions                        91199
special_conditions_at_site                     91199
carriageway_hazards                            91199
urban_or_rural_area                            91199
did_police_officer_attend_scene_of_accident    91199
trunk_road_flag                                91199
lsoa_of_accident_location                      91199
dtype: int64
```

Vehicle data 2020

In [13]:
```
# Vehicle data 2020
VEH2020
```

Out[1

	vehi	accic	accic	accic	vehi	vehi	towi	vehi	vehi	vehi	...	jourı	sex_	age_	age_	engi	prop	age_	gene	driv	d
6817	6817	2020	2020	0102	1	9	9	5	1	5	...	6	2	32	6	1968	2	6	AUD Q5	4	1
6817	6817	2020	2020	0102	1	9	0	4	2	6	...	2	1	45	7	1395	1	2	AUD A1	7	1
6817	6817	2020	2020	0102	1	9	0	18	-1	-1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
6817	6817	2020	2020	0102	1	8	0	18	1	5	...	1	1	44	7	1798	8	8	TOYC PRIU	2	1
6817	6817	2020	2020	0102	1	9	0	18	3	7	...	6	1	20	4	2993	2	4	BMW 4 SERII	-1	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8490	8490	2020	2020	9910	1	9	0	7	8	2	...	1	1	57	9	1968	2	2	AUD A5	7	1
8490	8490	2020	2020	9910	2	5	0	16	6	2	...	5	1	38	7	1301	1	2	KTM 1290 SUPE	9	2
8490	8490	2020	2020	9910	1	9	0	7	8	2	...	6	2	68	10	1995	2	1	BMW X3	5	1
8490	8490	2020	2020	9910	2	1	0	18	6	2	...	6	1	76	11	-1	-1	-1	-1	9	1
8490	8490	2020	2020	9910	1	9	0	1	8	4	...	6	1	39	7	999	1	2	FORI FOCU	7	1

In [16]:`VEH2020.count()`

Out[16]:

```
vehicle_index                          167375
accident_index                         167375
accident_year                          167375
accident_reference                     167375
vehicle_reference                      167375
vehicle_type                           167375
towing_and_articulation                167375
vehicle_manoeuvre                      167375
vehicle_direction_from                 167375
vehicle_direction_to                   167375
vehicle_location_restricted_lane       167375
junction_location                      167375
skidding_and_overturning               167375
hit_object_in_carriageway              167375
vehicle_leaving_carriageway            167375
hit_object_off_carriageway             167375
first_point_of_impact                  167375
vehicle_left_hand_drive                167375
journey_purpose_of_driver              167375
sex_of_driver                          167375
age_of_driver                          167375
age_band_of_driver                     167375
engine_capacity_cc                     167375
propulsion_code                        167375
age_of_vehicle                         167375
generic_make_model                     167375
driver_imd_decile                      167375
driver_home_area_type                  167375
dtype: int64
```

Casualty data 2020

In [14]:
```
# Casualty data 2020
CAS2020
```

Out[1

	casua	accid	accid	accid	vehic	casua	casua	sex_c	age_(	age_l	casua	pede	pede	car_p	bus_(	pede	casua	casua	ca
4847	4847	2020(	2020	0102	1	1	3	1	31	6	3	9	5	0	0	0	0	1	4
4847	4847	2020(	2020	0102	1	1	3	2	2	1	3	1	1	0	0	0	0	1	2
4847	4847	2020(	2020	0102	1	2	3	2	4	1	3	1	1	0	0	0	0	1	2
4847	4847	2020(	2020	0102	1	1	3	1	23	5	3	5	9	0	0	0	0	1	3
4847	4847	2020(	2020	0102	1	1	3	1	47	8	2	4	1	0	0	0	0	1	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6003	6003	2020!	2020	9910	2	1	1	1	11	3	2	0	0	0	0	0	1	1	2
6003	6003	2020!	2020	9910	1	1	3	2	63	9	3	10	1	0	0	0	0	1	1(
6003	6003	2020!	2020	9910	2	1	1	1	38	7	2	0	0	0	0	0	5	2	9
6003	6003	2020!	2020	9910	2	1	1	1	76	11	3	0	0	0	0	0	1	1	9
6003	6003	2020!	2020	9910	1	1	3	1	48	8	3	9	9	0	0	0	0	1	1

In [17]:
```
CAS2020.count()
```

Out[17]:

```
casualty_index                        115584
accident_index                        115584
accident_year                         115584
accident_reference                    115584
vehicle_reference                     115584
casualty_reference                    115584
casualty_class                        115584
sex_of_casualty                       115584
age_of_casualty                       115584
age_band_of_casualty                  115584
casualty_severity                     115584
pedestrian_location                   115584
pedestrian_movement                   115584
car_passenger                         115584
bus_or_coach_passenger                115584
pedestrian_road_maintenance_worker    115584
casualty_type                         115584
casualty_home_area_type               115584
casualty_imd_decile                   115584
dtype: int64
```

1. Are there significant hours of the day, and days of the week, on which accidents occur?

Significant hours of the day on which accidents occur

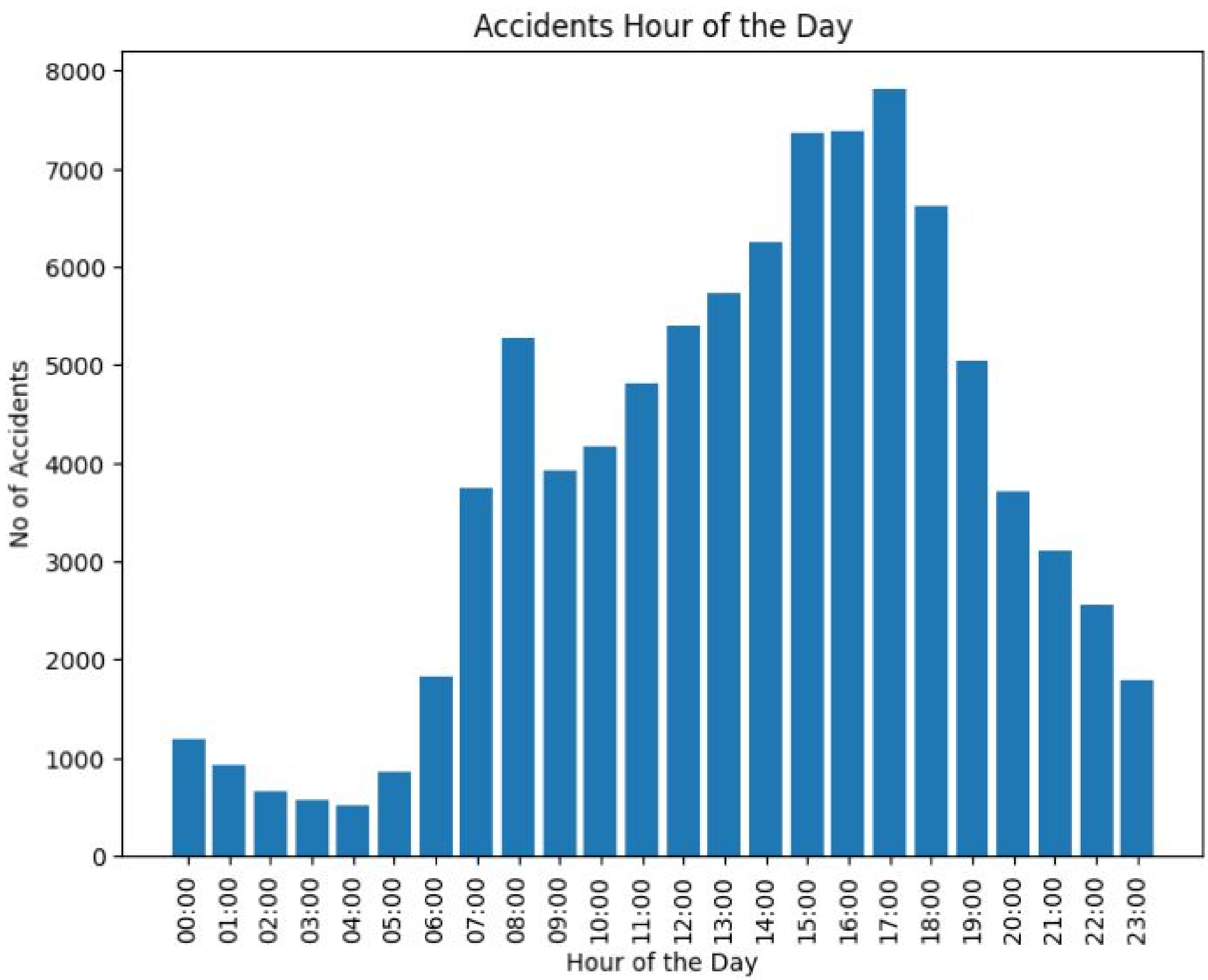
In [18]:
```
# Significant hours of the day on which accident occur

ACC2020.groupby("time")["accident_index"].count().sort_values(ascending = F
```

Out[18]:

```
time
17:00    862
16:00    785
15:00    774
17:30    746
18:00    739
        ...
04:39      1
04:58      1
04:31      1
03:36      1
04:33      1
Name: accident_index, Length: 1438, dtype: int64
```

The above shows that the significant hours of the day on which accidents occur and peaking at 17:00 giving a total of 862 accidents in the road traffic accidents data for 2020

In [19]:
```
ACC2020['hour_of_day'] = pd.to_datetime(ACC2020['time']).dt.hour

# Count the number of accidents in each hour of the day
Accidents_hours = ACC2020['hour_of_day'].value_counts().sort_index()

# Plot the data
plt.figure(figsize=(8, 6))
plt.bar(Accidents_hours.index, Accidents_hours.values)
plt.xlabel('Hour of the Day')
plt.ylabel('No of Accidents')
plt.title('Accidents Hour of the Day')
plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)
plt.show()
```

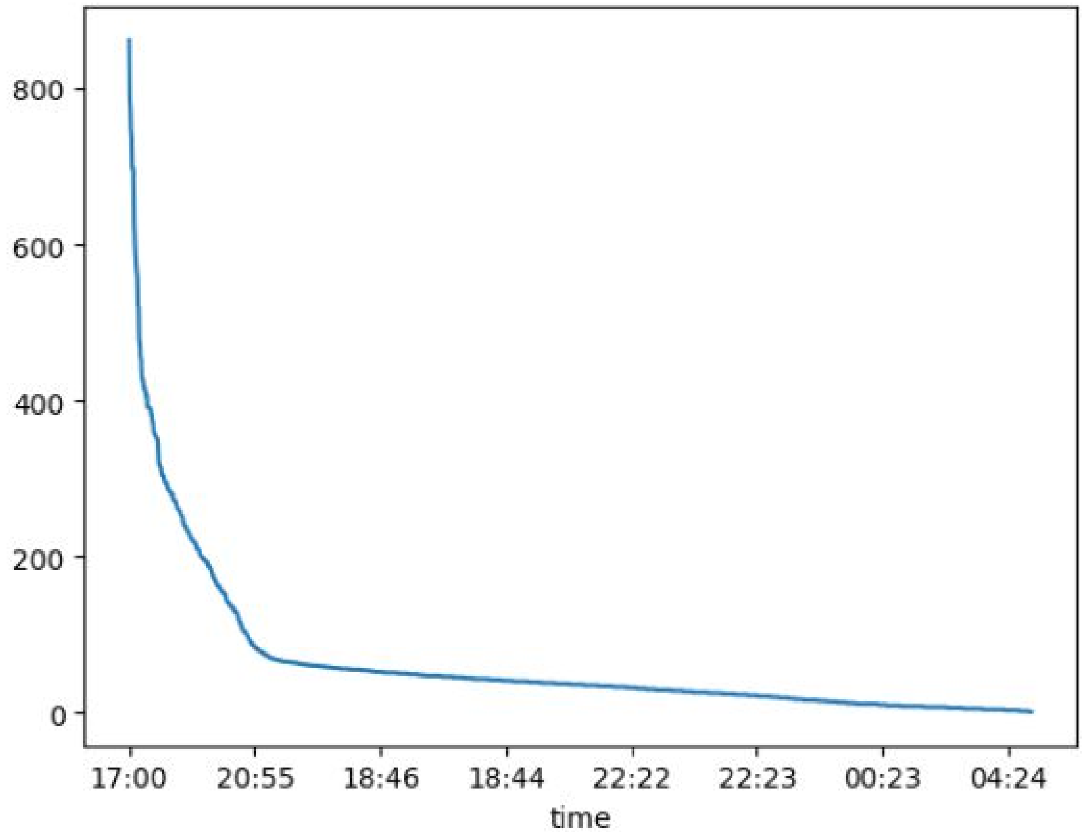


The above gragh shows insights of the number of accident happening hourly from the accident data

In [20]:
```
ACC2020.groupby("time")["accident_index"].count().sort_values(ascending = F
```

Out[20]:

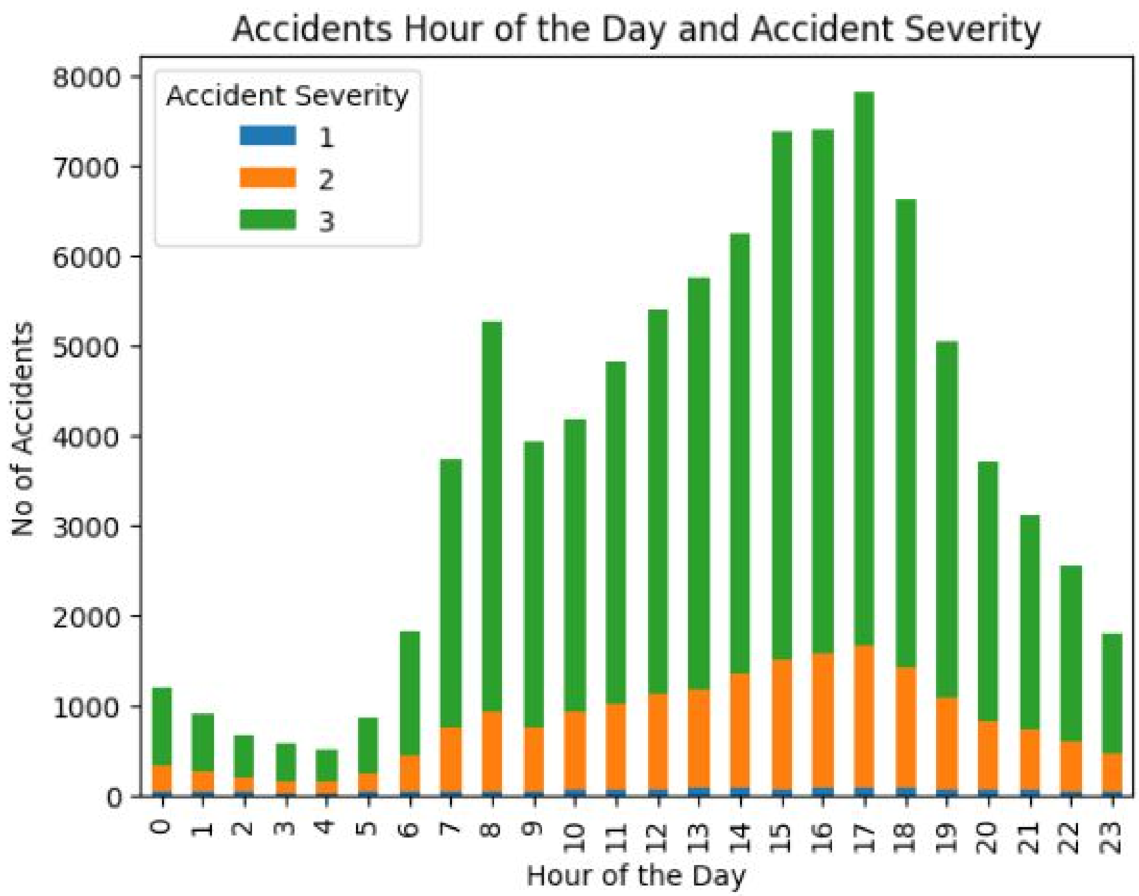
```
<Axes: xlabel='time'>
```



Accidents data by hour and Accident severity

In [21]:
```
# Accidents data by hour and Accident severity
ACC2020['hour_of_day'] = pd.to_datetime(ACC2020['time']).dt.hour
pivot_table = pd.pivot_table(ACC2020, values='accident_index', index='hour_

# Plotting the data
plt.figure(figsize=(8, 6))
pivot_table.plot(kind='bar', stacked=True)
plt.xlabel('Hour of the Day')
plt.ylabel('No of Accidents')
plt.title('Accidents Hour of the Day and Accident Severity')
plt.xticks(rotation=90)
plt.legend(title='Accident Severity')
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```

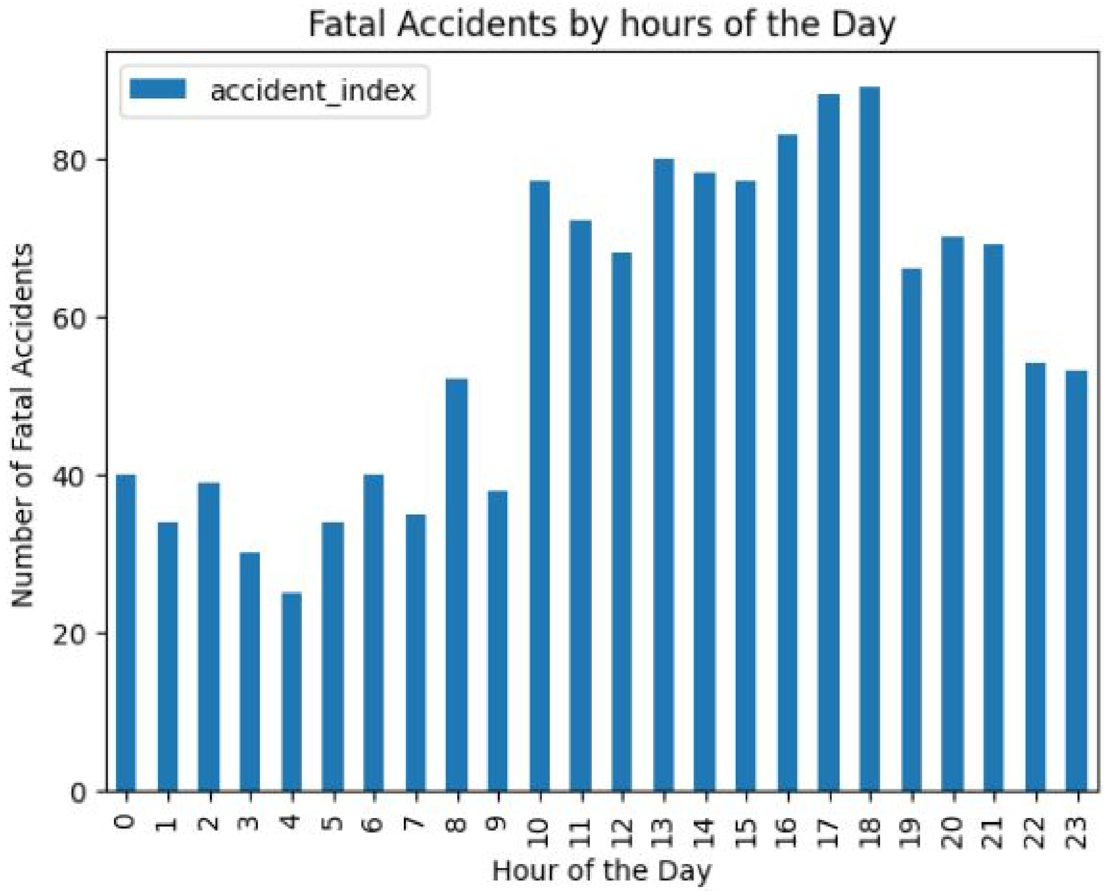


The above plot shows the relationship between the Accident hours of the day and the Accident Severity. 1 - Fatal, 2 -Serious, 3 - Slight.

fatal accident severity

In [22]:
```
# Filter the DataFrame to include only accidents with accident_severity = 1
Accidents_severity_1 = ACC2020[ACC2020['accident_severity'] == 1].copy()

# Extract the hour of the day from the 'time' column
Accidents_severity_1['hour_of_day'] = pd.to_datetime(Accidents_severity_1['

# Create a pivot table to count accidents with severity 1 by hour
pivot_table = pd.pivot_table(Accidents_severity_1, values='accident_index',

# Plot the data
plt.figure(figsize=(8, 6))
pivot_table.plot(kind='bar')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Fatal Accidents')
plt.title('Fatal Accidents by hours of the Day')
plt.xticks(rotation=90)
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



In [23]:
```
#Creating a dataframe

ACCQL = ACC2020.groupby("time")["accident_index"].count().reset_index()
```

In [24]:
```
# Renaming the datafrmae

ACCQL.rename(columns = {"accident_index": "number_of_accidents"}, inplace =
```

In [25]:
```
ACCQL
```

Out[2

	time	number_of_acciden
0	00:00	36
1	00:01	91
2	00:02	16
3	00:03	15
4	00:04	23
...	...	...
1433	23:55	40
1434	23:56	14
1435	23:57	16
1436	23:58	9
1437	23:59	14

Significant days of the week, on which accidents occur

Using data from the Accident index table

In [26]:
```
ACC2020["date"] = pd.to_datetime(ACC2020["date"], dayfirst = True, format =
```

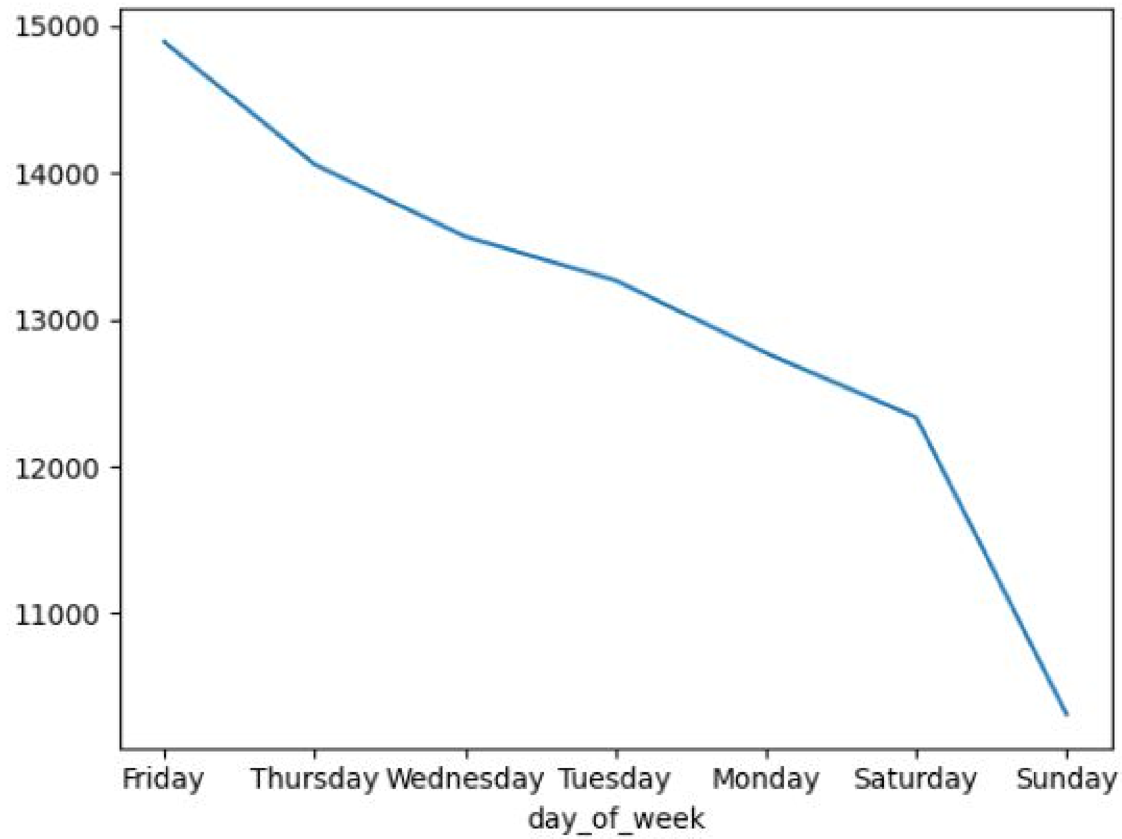
In [27]:
```
ACC2020.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 91199 entries, 370153 to 461351
Data columns (total 37 columns):
 #   Column                                      Non-Null Count  Dtyp
e
---  ------                                      --------------  -----

 0   accident_index                              91199 non-null  objec
t
 1   accident_year                               91199 non-null  int6
4
 2   accident_reference                          91199 non-null  objec
t
 3   location_easting_osgr                       91185 non-null  float6
4
 4   location_northing_osgr                      91185 non-null  float6
4
 5   longitude                                   91185 non-null  float6
4
 6   latitude                                    91185 non-null  float6
4
 7   police_force                                91199 non-null  int6
4
 8   accident_severity                           91199 non-null  int6
4
 9   number_of_vehicles                          91199 non-null  int6
4
 10  number_of_casualties                        91199 non-null  int6
4
 11  date                                        91199 non-null  datetime64[n
s]
 12  day_of_week                                 91199 non-null  int6
4
 13  time                                        91199 non-null  objec
t
 14  local_authority_district                    91199 non-null  int6
4
 15  local_authority_ons_district                91199 non-null  objec
t
 16  local_authority_highway                     91199 non-null  objec
t
 17  first_road_class                            91199 non-null  int6
4
 18  first_road_number                           91199 non-null  int6
4
 19  road_type                                   91199 non-null  int6
4
 20  speed_limit                                 91199 non-null  int6
4
 21  junction_detail                             91199 non-null  int6
4
 22  junction_control                            91199 non-null  int6
```

In [28]:
```
ACC2020["day_of_week"]= ACC2020["date"].dt.day_name()
```
In [29]:
```
ACC2020["day_of_week"]
```
Out[29]:
```
370153      Tuesday
370154       Monday
370155    Wednesday
370156    Wednesday
370157    Wednesday
            ...
461347    Wednesday
461348       Friday
461349    Wednesday
461350      Tuesday
461351      Tuesday
Name: day_of_week, Length: 91199, dtype: object
```
In [30]:
```
ACC2020.groupby("day_of_week")["accident_index"].count().sort_values(ascend
```
Out[30]:
```
day_of_week
Friday       14889
Thursday     14056
Wednesday    13564
Tuesday      13267
Monday       12772
Saturday     12336
Sunday       10315
Name: accident_index, dtype: int64
```
In [31]:
```
# Showing significant days of the week on which highest accident occur

ACC2020.groupby("day_of_week")["accident_index"].count().sort_values(ascend
```

Out[31]:

```
<Axes: xlabel='day_of_week'>
```



The above plot shows the significant days of the week on which highest accident occur

In [ ]:

2. For motorbikes, are there significant hours of the day, and days of the week, on which accidents occur? We suggest a focus on: Motorcycle 125cc and under, Motorcycle over 125cc and up to 500cc, and Motorcycle over 500cc

In [32]:
```
#Showing the unique vehicle types from the VEH2020 data

VEH2020.vehicle_type.unique()
```

Out[32]:

```
array([ 9,  8,  3,  2, 11,  1, 90, 19,  4,  5, 21, 97, 20, 98, 10, 17, 23,
       22, 18, 16], dtype=int64)
```

Accidents data for Motorcycle 125cc and under

In [33]:
```
motorbikes = pd.merge(ACC2020, VEH2020, on='accident_index', how='inner')
```

In [34]:
```
motorbikes
```

Out[3

	acci	acci	acci	loca	loca	long	latit	polic	acci	num	...	jour	sex_	age_	age_	engi	prop	age_	gene	driv	d
0	2020	2020	0102	5213	1751	-0.254	51.4	1	3	1	...	6	2	32	6	1968	2	6	AUD Q5	4	1
1	2020	2020	0102	5293	1762	-0.139	51.4	1	3	1	...	2	1	45	7	1395	1	2	AUD A1	7	1
2	2020	2020	0102	5264	1827	-0.178	51.5	1	3	1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
3	2020	2020	0102	5386	1843	-0.00	51.5	1	2	1	...	1	1	44	7	1798	8	8	TOY PRIU	2	1
4	2020	2020	0102	5293	1812	-0.137	51.5	1	3	1	...	6	1	20	4	2993	2	4	BMW 4 SERI	-1	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1673	2020	2020	9910	3836	8106	-2.27	57.1	99	2	2	...	1	1	57	9	1968	2	2	AUD A5	7	1
1673	2020	2020	9910	3836	8106	-2.27	57.1	99	2	2	...	5	1	38	7	1301	1	2	KTM 1290 SUPE	9	2
1673	2020	2020	9910	2771	6748	-3.968	55.9	99	3	2	...	6	2	68	10	1995	2	1	BMW X3	5	1
1673	2020	2020	9910	2771	6748	-3.968	55.9	99	3	2	...	6	1	76	11	-1	-1	-1	-1	9	1
1673	2020	2020	9910	2404	6819	-4.56	56.0	99	3	1	...	6	1	39	7	999	1	2	FOR FOC	7	1

In [35]:
```
VEH2020_CAT2 = motorbikes[motorbikes["vehicle_type"].isin([2, 3])]
VEH2020_CAT2
```

Out[3

	acci	acci	acci	loca	loca	long	latit	polic	acci	num	...	jour	sex_	age_	age_	engi	prop	age_	gene	driv	d
12	2020	2020	0102	5297	1923	- 0.127	51.6	1	3	2	...	6	1	37	7	114	1	5	YAM XC11	8	1
32	2020	2020	0102	5314	1745	- 0.109	51.4	1	2	2	...	6	1	19	4	-1	-1	-1	-1	5	1
33	2020	2020	0102	5310	1763	- 0.114	51.4	1	3	1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
36	2020	2020	0102	5284	1799	- 0.150	51.5	1	3	2	...	6	1	20	4	125	1	4	PEU TWE 125	6	1
58	2020	2020	0102	5184	1852	- 0.292	51.5	1	3	1	...	6	1	20	4	125	1	0	HON GLR 125	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1670	2020	2020	9910	3406	6740	- 2.95	55.9	99	2	2	...	5	1	18	4	-1	-1	-1	-1	2	2
1671	2020	2020	9910	2767	6670	- 3.97	55.8	99	2	2	...	2	1	17	4	125	1	2	KEEV RK 125	3	1
1671	2020	2020	9910	2506	6580	- 4.38	55.7	99	3	2	...	6	1	47	8	124	1	17	-1	5	1
1672	2020	2020	9910	2681	6566	- 4.10	55.7	99	3	2	...	2	1	61	9	124	1	4	-1	3	1
1673	2020	2020	9910	3116	6837	- 3.41	56.0	99	2	2	...	1	1	35	6	125	1	3	-1	5	1

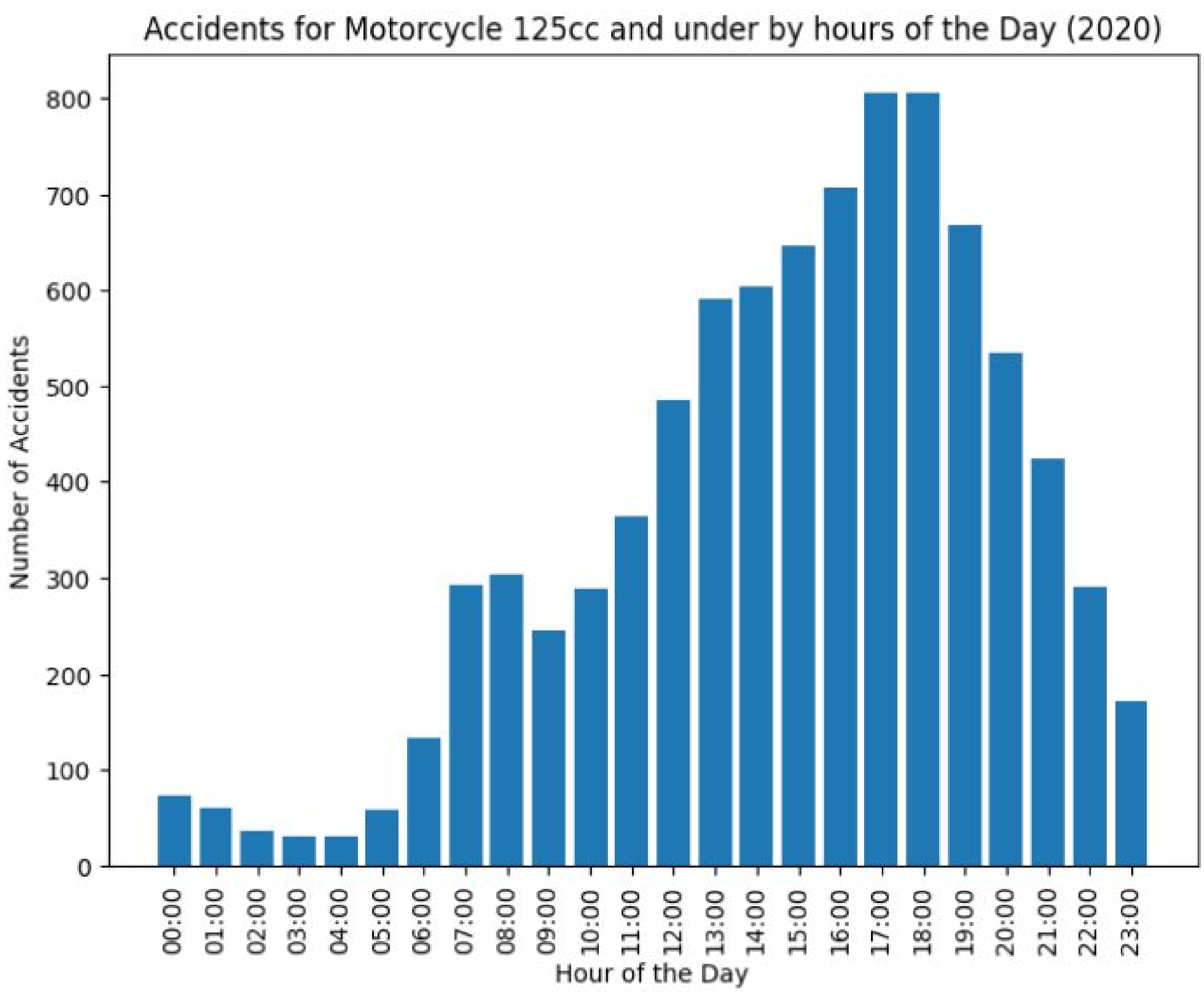
In [36]:
```
# significant hours of the day on which accident occur

VEH2020_CAT2.groupby("time")["accident_index"].count().sort_values(ascendin
```

Out[36]:

```
time
17:00    90
17:30    81
19:00    80
15:30    79
18:30    77
         ..
08:59     1
09:01     1
02:16     1
09:02     1
09:24     1
Name: accident_index, Length: 1166, dtype: int64
```

In [38]:
```
# Filter the data for Motorcycle 125cc and under (Vehicle Type 02 or 03)
```

```
motorcycle_125cc_under = motorbikes[motorbikes['vehicle_type'].isin([2, 3])

# Extract the hour of the day from the 'time' column
motorcycle_125cc_under.loc[:, 'hour_of_day'] = pd.to_datetime(motorcycle_12

# Count the number of accidents in each hour of the day
hourly_accidents_125cc_under = motorcycle_125cc_under['hour_of_day'].value_

# Plot the data
plt.figure(figsize=(8, 6))
plt.bar(hourly_accidents_125cc_under.index, hourly_accidents_125cc_under.va
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.title('Accidents for Motorcycle 125cc and under by hours of the Day (20
plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)
plt.show()
```



In [ ]:

Accident data for Motorcycle over 125cc and up to 500cc

```
VEH2020_CAT4 = motorbikes[motorbikes["vehicle_type"].isin([4])]
```

In [39]:
```
VEH2020_CAT4
```

Out[3

	accid	accid	accid	locat	locat	long	latitu	polic	accid	num	...	journ	sex_	age_	age_	engi	prop	age_	gene	drive	d
92	2020	2020	0102	5401	1903	0.022	51.59	1	3	2	...	6	1	39	7	125	1	7	HON SH12	1	1
106	2020	2020	0102	5309	1710	-0.118	51.42	1	3	1	...	6	1	33	6	125	1	3	HON WW1	5	1
618	2020	2020	0102	5377	1805	-0.016	51.50	1	3	2	...	2	1	44	7	395	1	10	-1	5	1
722	2020	2020	0102	5257	1750	-0.190	51.46	1	3	3	...	6	1	26	6	499	1	16	-1	3	1
750	2020	2020	0102	5255	1767	-0.193	51.47	1	3	3	...	6	1	18	4	-1	-1	-1	PIAG MP3	3	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1663	2020	2020	9909	3394	7341	-2.985	56.49	99	2	1	...	6	1	21	5	-1	-1	-1	-1	4	1
1667	2020	2020	9910	3615	8624	-2.646	57.64	99	2	3	...	5	1	29	6	249	1	16	-1	3	1
1668	2020	2020	9910	3222	6720	-3.246	55.93	99	2	2	...	5	2	27	6	125	1	1	-1	7	1
1670	2020	2020	9910	3243	6728	-3.212	55.94	99	3	2	...	1	1	38	7	249	1	9	-1	7	1
1673	2020	2020	9910	3180	7452	-3.336	56.59	99	3	2	...	5	1	48	8	250	1	5	-1	2	1

significant hours of the day on which accidents occur

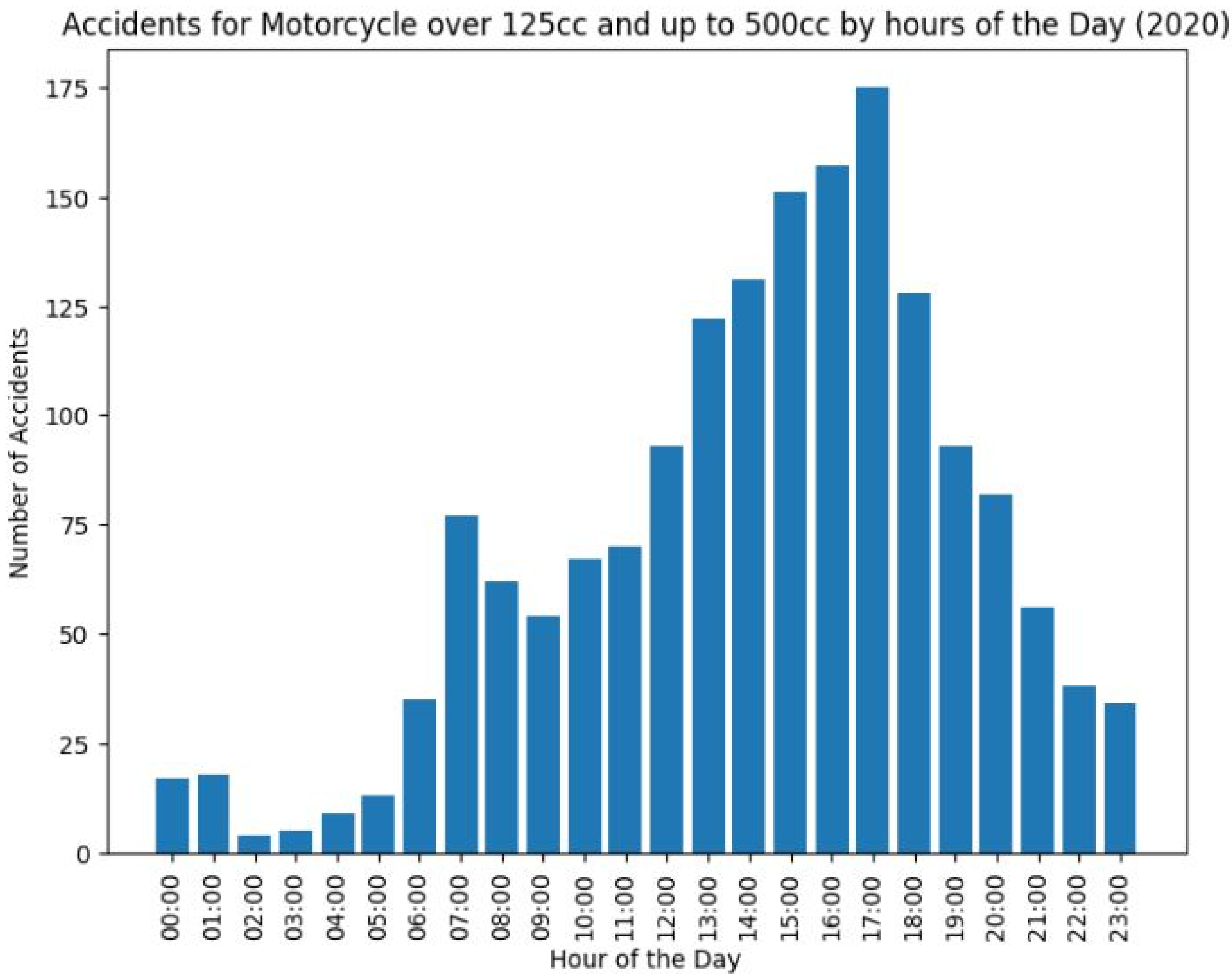
In [40]:
```
# significant hours of the day on which accident occur

VEH2020_CAT4.groupby("time")["accident_index"].count().sort_values(ascendin
```

Out[40]:

```
time
17:00    23
16:00    18
16:30    17
15:30    17
15:00    16
         ..
12:38     1
12:34     1
12:31     1
12:28     1
23:50     1
Name: accident_index, Length: 685, dtype: int64
```

In [41]:
```
# Filter the data for Motorcycle over 125cc and up to 500cc (Vehicle Type 0
```

```
motorcycle_125cc_over = motorbikes[motorbikes['vehicle_type'].isin([4])].co

# Extract the hour of the day from the 'time' column
motorcycle_125cc_over.loc[:, 'hour_of_day'] = pd.to_datetime(motorcycle_125

# Count the number of accidents in each hour of the day
hourly_accidents_125cc_over = motorcycle_125cc_over['hour_of_day'].value_co

# Plot the data
plt.figure(figsize=(8, 6))
plt.bar(hourly_accidents_125cc_over.index, hourly_accidents_125cc_over.valu
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.title('Accidents for Motorcycle over 125cc and up to 500cc by hours of
plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)
plt.show()
```



Accident data for Motorcycle over 500cc

In [42]:
```
VEH2020_CAT5 = motorbikes[motorbikes["vehicle_type"].isin([5])]
VEH2020_CAT5
```

Out[4

	acci	acci	acci	locat	locat	long	latit	polic	acci	num	...	jour	sex_	age_	age_	engi	prop	age_	gene	driv	d
119	2020	2020	0102	5245	1786	-0.207	51.49	1	2	2	...	6	1	35	6	600	1	8	YAM XJ6	2	1
334	2020	2020	0102	5307	1874	-0.115	51.57	1	3	2	...	6	1	48	8	1200	1	17	HARI - DAVI MOD MISS	2	1
381	2020	2020	0102	5152	1830	-0.339	51.53	1	3	4	...	2	1	40	7	647	1	2	-1	6	1
402	2020	2020	0102	5307	1793	-0.117	51.49	1	3	2	...	6	1	31	6	865	1	15	TRIU BON	4	1
450	2020	2020	0102	5485	1783	0.138	51.48	1	3	2	...	6	1	18	4	847	1	3	YAM MT09	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1667	2020	2020	9910	3615	8624	-2.646	57.64	99	2	3	...	5	1	30	6	599	1	19	-1	6	2
1667	2020	2020	9910	3262	6697	-3.182	55.91	99	2	1	...	5	1	53	8	649	1	3	KAW KLE6	7	1
1668	2020	2020	9910	3385	7318	-2.999	56.47	99	2	2	...	2	1	50	8	600	1	20	KAW MOD MISS	1	1
1671	2020	2020	9910	2761	6638	-3.979	55.85	99	3	2	...	5	1	48	8	649	1	18	-1	8	1
1673	2020	2020	9910	3836	8106	-2.271	57.18	99	2	2	...	5	1	38	7	1301	1	2	KTM 1290 SUPE	9	2

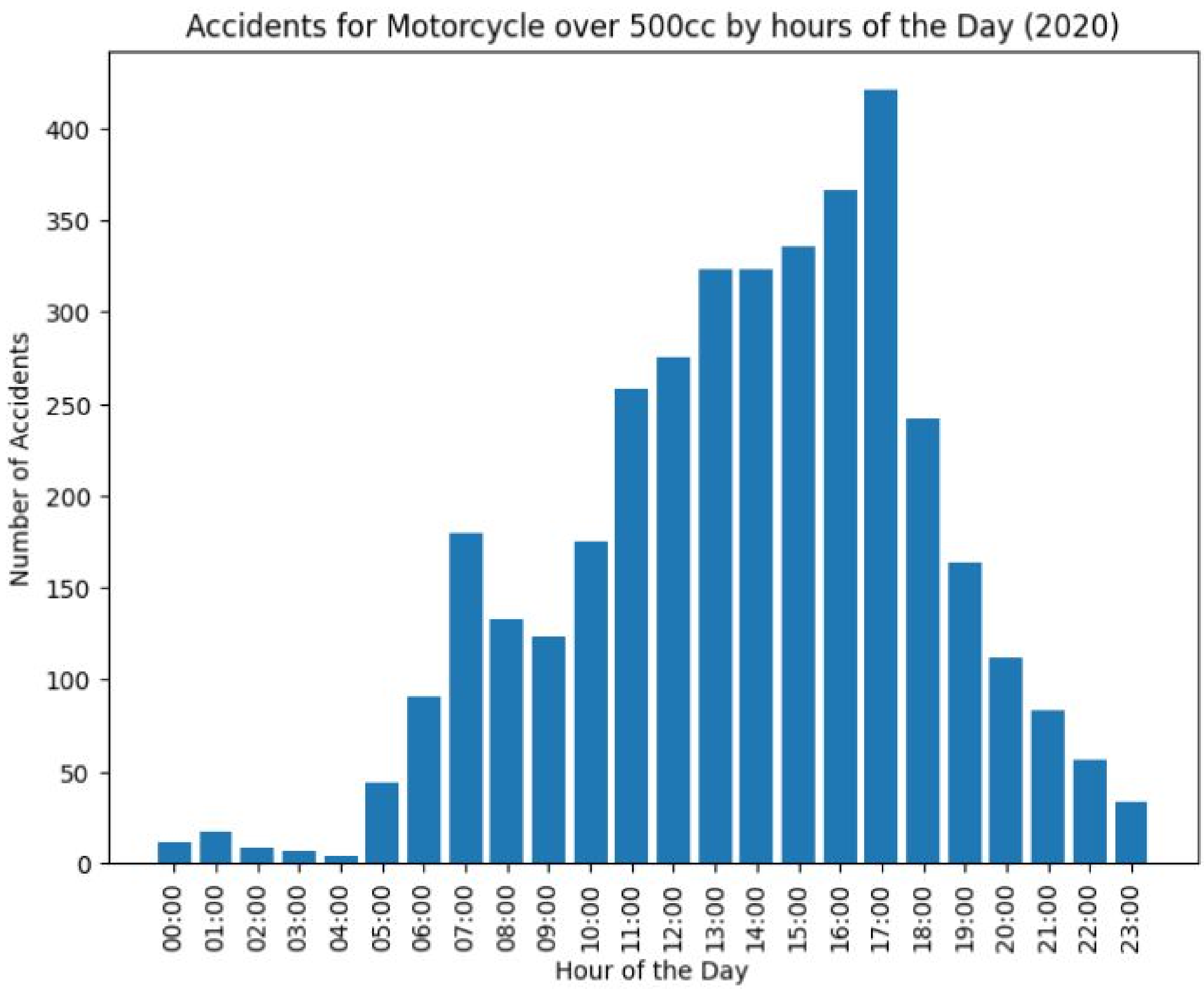
significant hours of the day on which accidents occur

In [43]:
```
# significant hours of the day on which accident occur

VEH2020_CAT5.groupby("time")["accident_index"].count().sort_values(ascendin
```

Out[43]:

```
time
17:30    43
17:00    35
14:00    34
15:00    31
16:00    31
         ..
11:31     1
11:29     1
11:26     1
11:17     1
23:53     1
Name: accident_index, Length: 913, dtype: int64
```

In [44]:
```
# Filter the data for Motorcycle over 500cc (Vehicle Type 05)
motorcycle_500cc_over = motorbikes[motorbikes['vehicle_type'].isin([5])].co

# Extract the hour of the day from the 'time' column
motorcycle_500cc_over.loc[:, 'hour_of_day'] = pd.to_datetime(motorcycle_500

# Count the number of accidents in each hour of the day
hourly_accidents_500cc_over = motorcycle_500cc_over['hour_of_day'].value_co

# Plot the data
plt.figure(figsize=(8, 6))
plt.bar(hourly_accidents_500cc_over.index, hourly_accidents_500cc_over.valu
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.title('Accidents for Motorcycle over 500cc by hours of the Day (2020)')
plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)
plt.show()
```



Accident data for Motorcycle with unknown cc

In [45]:
```
VEH2020_CAT97 = motorbikes[motorbikes["vehicle_type"].isin([97])]
VEH2020_CAT97
```

Out[4

	acci	acci	acci	locat	locat	long	latit	polic	acci	num	...	jour	sex_	age_	age_	engi	prop	age_	gene	driv	d
204	2020	2020	0102	5247	1802	-0.20	51.5	1	3	3	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
867	2020	2020	0102	5235	1780	-0.22	51.4	1	3	1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
1331	2020	2020	0102	5129	1803	-0.37	51.5	1	3	1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
1736	2020	2020	0102	5362	1846	-0.03	51.5	1	3	2	...	6	3	21	5	108	1	6	-1	4	1
1935	2020	2020	0102	5312	1784	-0.11	51.4	1	3	1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1641	2020	2020	9909	1803	8533	-5.67	57.5	99	3	1	...	5	1	65	9	1584	1	13	-1	5	3
1644	2020	2020	9909	2658	6664	-4.14	55.8	99	2	1	...	6	1	26	6	-1	-1	-1	-1	9	1
1645	2020	2020	9909	3384	7341	-3.00	56.4	99	3	2	...	6	3	20	4	-1	-1	-1	-1	-1	-
1650	2020	2020	9909	2580	6662	-4.27	55.8	99	3	2	...	1	1	19	4	-1	-1	-1	-1	-1	-
1664	2020	2020	9909	2588	6645	-4.25	55.8	99	2	1	...	1	1	26	6	-1	-1	-1	-1	4	1

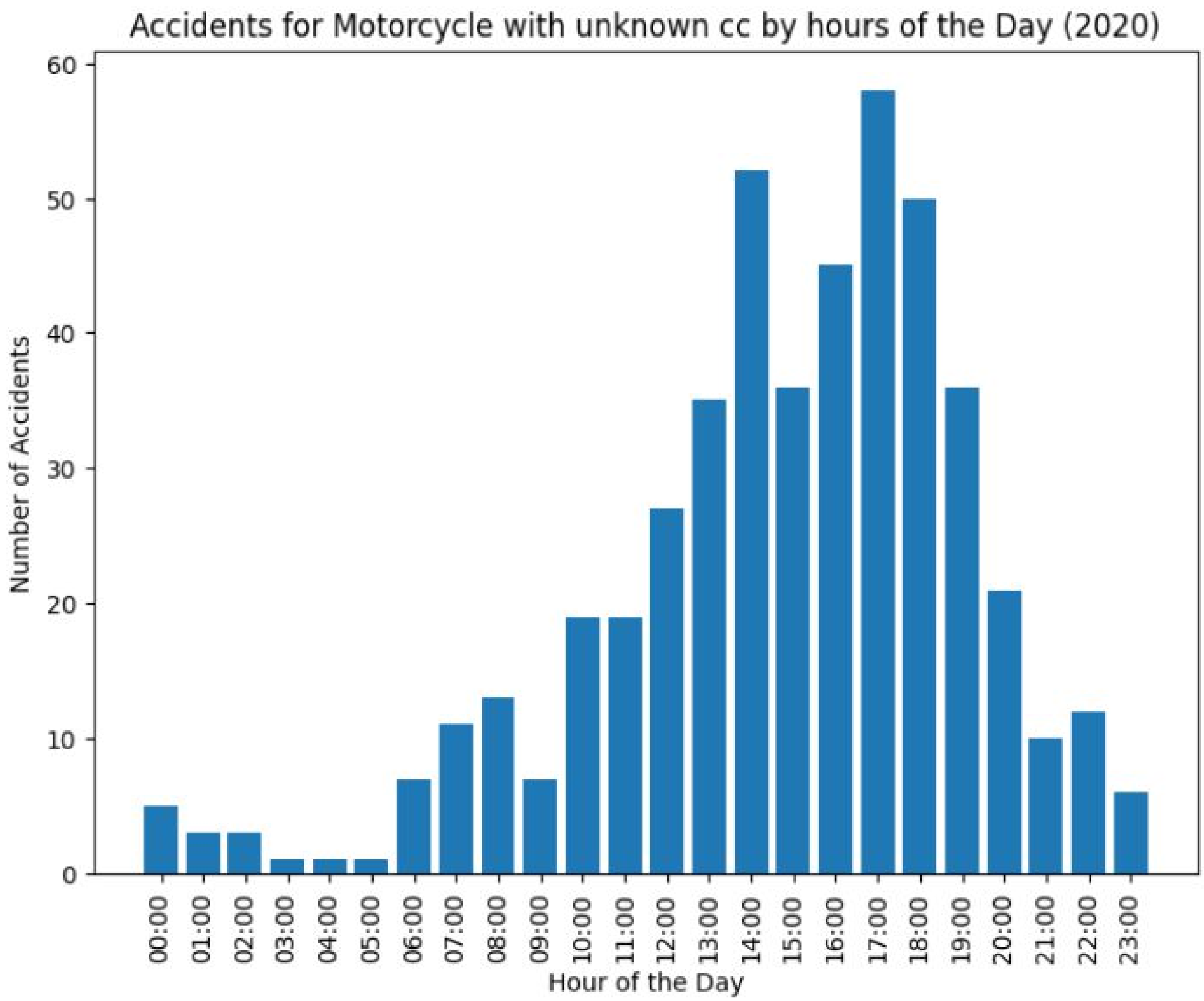
significant hours of the day on which accidents occur

In [46]:
```
# significant hours of the day on which accident occur

VEH2020_CAT97.groupby("time")["accident_index"].count().sort_values(ascendi
```

Out[46]:
```
time
18:00    9
17:00    8
17:05    7
14:30    7
14:00    7
        ..
14:52    1
14:54    1
14:58    1
14:59    1
23:56    1
Name: accident_index, Length: 295, dtype: int64
```

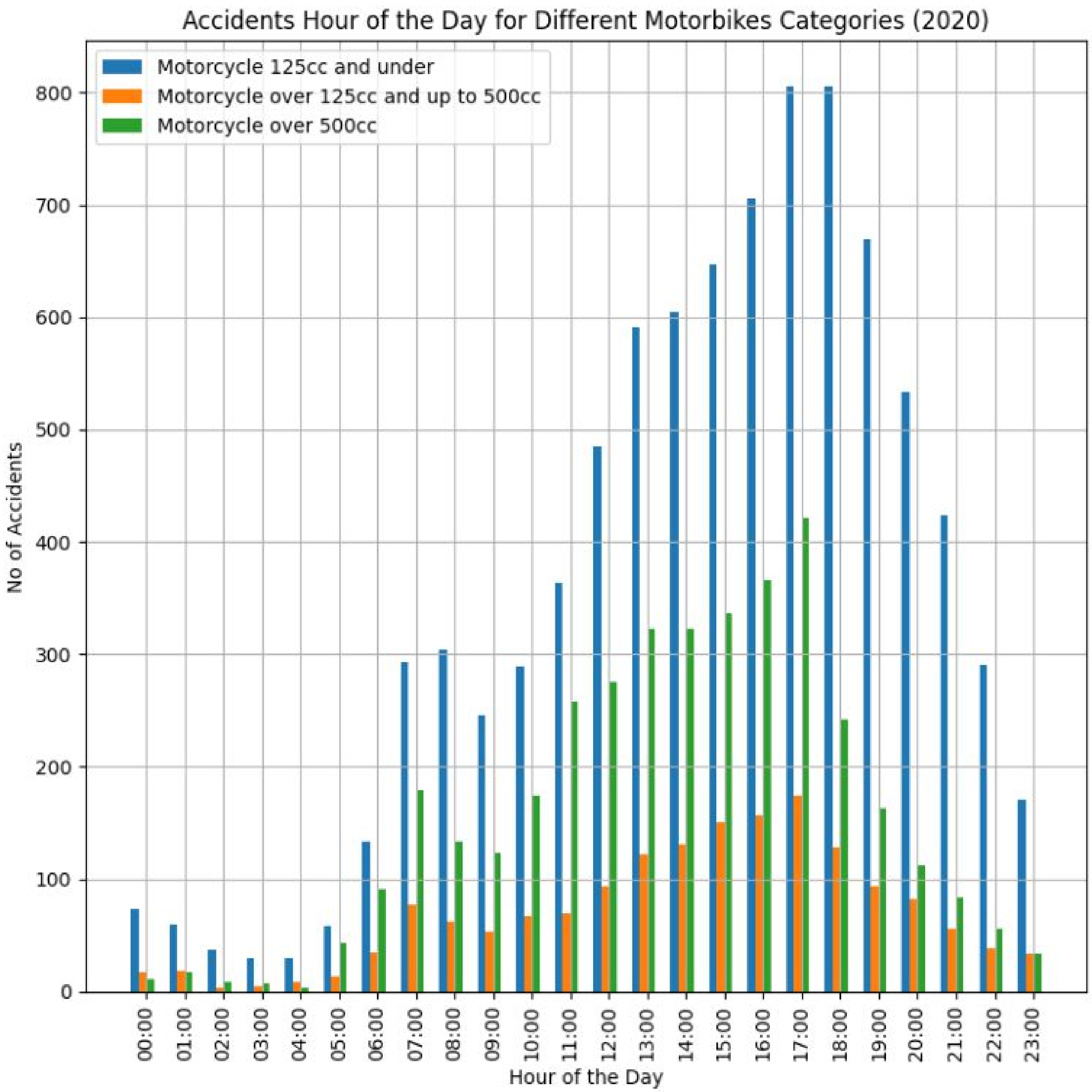
In [47]:
```
# Filter the data for Motorcycle with unknown cc (Vehicle Type 97)
motorcycle_unknowncc = motorbikes[motorbikes['vehicle_type'].isin([97])].co
```

```
# Extract the hour of the day from the 'time' column
motorcycle_unknowncc.loc[:, 'hour_of_day'] = pd.to_datetime(motorcycle_unkn

# Count the number of accidents in each hour of the day
hourly_accidents_unknowncc = motorcycle_unknowncc['hour_of_day'].value_coun

# Plot the data
plt.figure(figsize=(8, 6))
plt.bar(hourly_accidents_unknowncc.index, hourly_accidents_unknowncc.values
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Accidents')
plt.title('Accidents for Motorcycle with unknown cc by hours of the Day (20
plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)
plt.show()
```



In [48]:
```
# plotting accidents by hour for each motorcycle category
def plot_hourly_accidents(motorbikes, vehicle_types, title, bar_width, offs
```

```
    motorbikes_cat = motorbikes[motorbikes['vehicle_type'].isin(vehicle_typ
    motorbikes_cat['hour_of_day'] = pd.to_datetime(motorbikes_cat['time']).

    # Counting the no of accidents in hour of the day
    hourly_accidents = motorbikes_cat['hour_of_day'].value_counts().sort_in

    # Plotting the gragh
    plt.bar(hourly_accidents.index + offset, hourly_accidents.values, width
    plt.xlabel('Hour of the Day')
    plt.ylabel('No of Accidents')
    plt.title('Accidents Hour of the Day for Different Motorbikes Categorie
    plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)

# plot for motorbikes cat
plt.figure(figsize=(8, 8))

# Set the width of the bars
bar_width = 0.2

# Plot for Motorcycle 125cc and under (Vehicle Type 02 or 03)
plot_hourly_accidents(VEH2020_CAT2, [2, 3], 'Motorcycle 125cc and under', b

# Plot for Motorcycle over 125cc and up to 500cc (Vehicle Type 04)
plot_hourly_accidents(VEH2020_CAT4, [4], 'Motorcycle over 125cc and up to 5

# Plot for Motorcycle over 500cc (Vehicle Type 05)
plot_hourly_accidents(VEH2020_CAT5, [5], 'Motorcycle over 500cc', bar_width


plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [49]:
```
#Creating a dataframe

motorbikes_catQL = motorbikes.groupby("time")["accident_index"].count().res
```
In [50]:
```
# Renaming the datafrmae

motorbikes_catQL.rename(columns = {"accident_index": "number_of_motorcycles
```
In [51]:
```
motorbikes_catQL
```

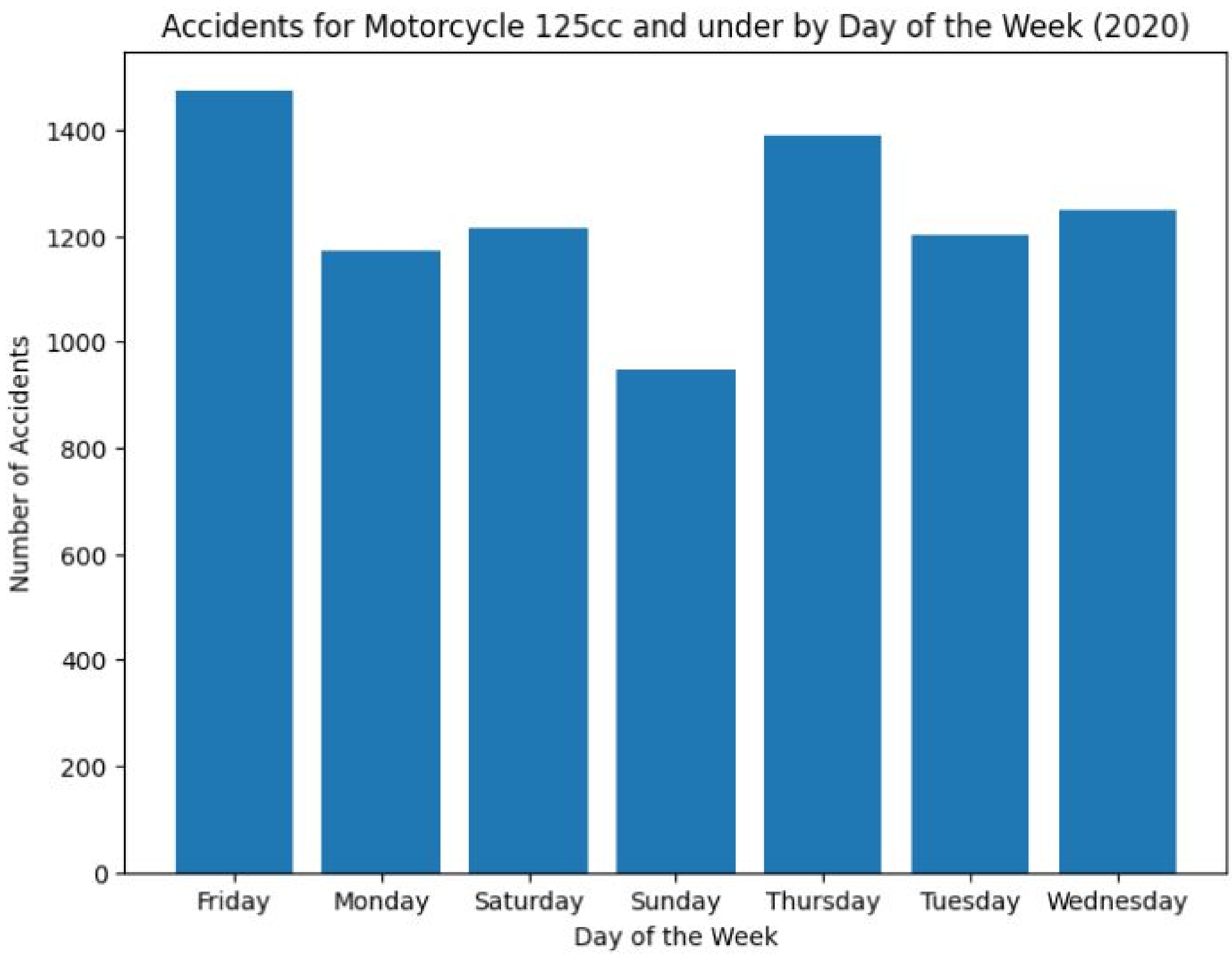
Out[5

		time	number_of_motorcycl
0	00:00	65	
1	00:01	149	
2	00:02	25	
3	00:03	24	
4	00:04	33	
...	...	...	
1433	23:55	68	
1434	23:56	20	
1435	23:57	29	
1436	23:58	11	
1437	23:59	29	

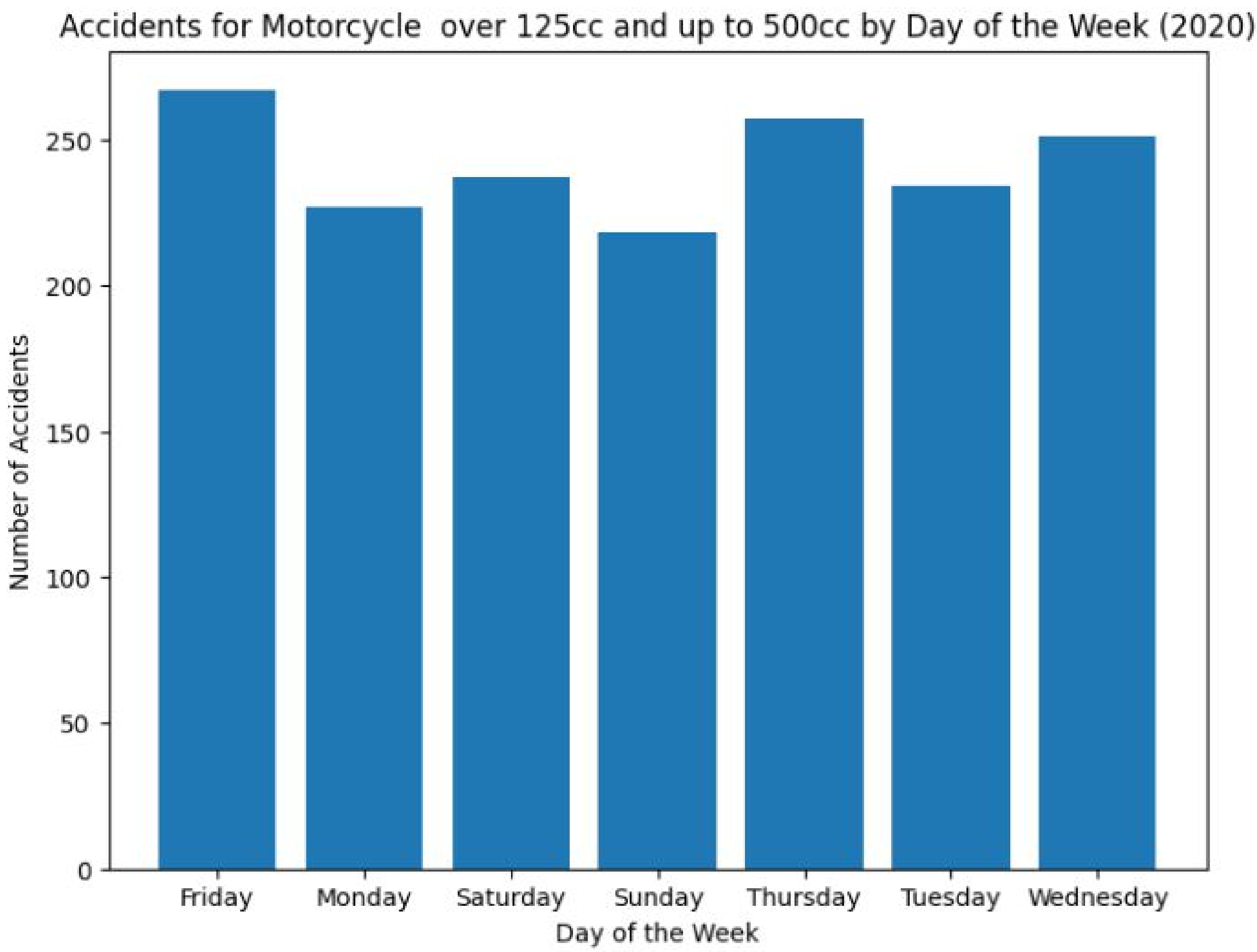
Significant days of the week, on which accidents occur by the vehicle group

In [91]:
```
# Define the function for analyzing accidents by day of the week
def accidents_by_day_of_week(vehicle_category, title):
    # Count the number of accidents on each day of the week
    daily_accidents = vehicle_category['day_of_week'].value_counts().sort_i


    # Define the order of days of the week (Sunday = 1, Saturday = 7)
    days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday

    # Plot the data
    plt.figure(figsize=(8, 6))
    plt.bar(daily_accidents.index, daily_accidents.values)
    plt.xlabel('Day of the Week')
    plt.ylabel('Number of Accidents')
    plt.title(f'Accidents for {title} by Day of the Week (2020)')
    plt.xticks(days)
    plt.show()
```

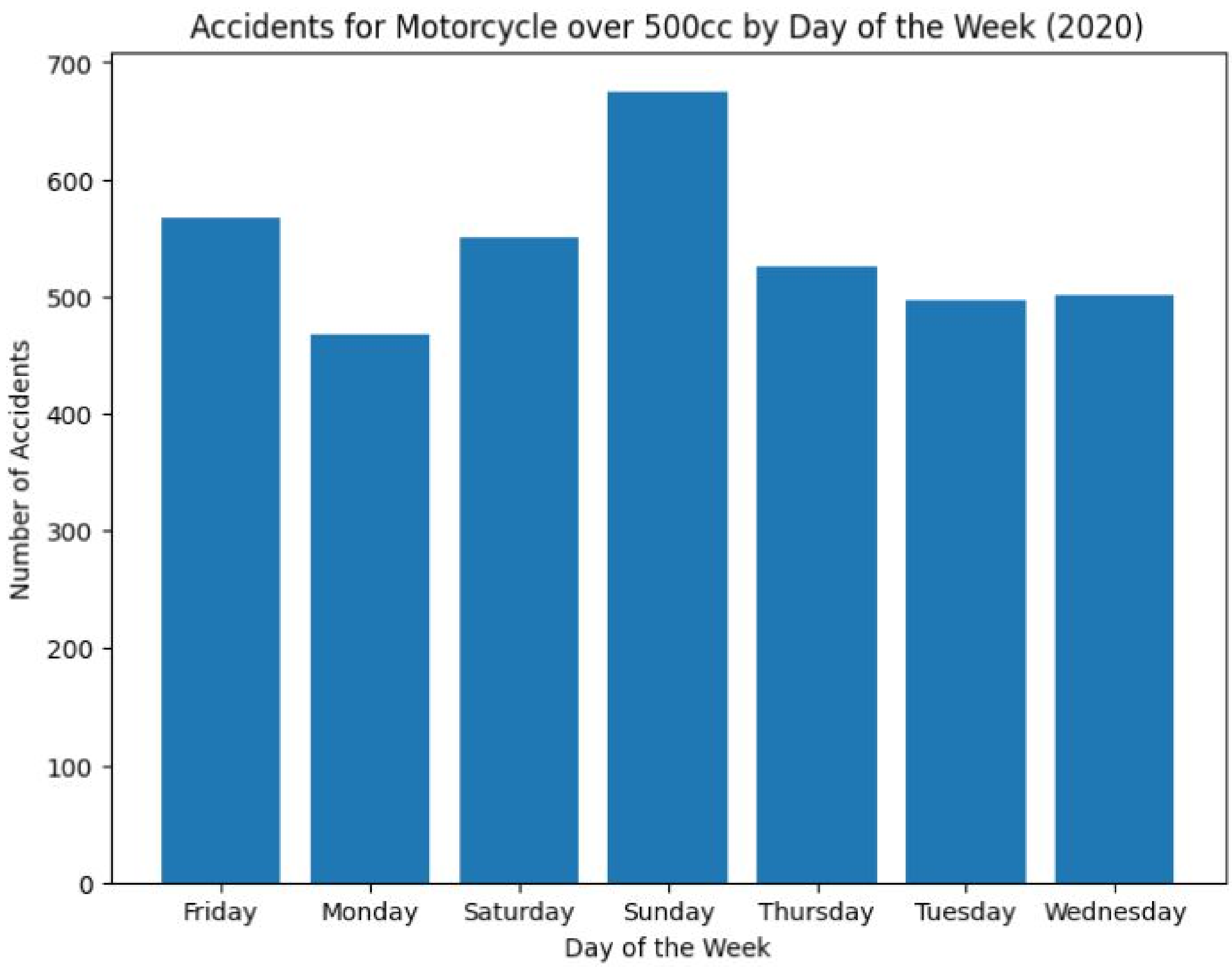
In [92]:
```
accidents_by_day_of_week(motorcycle_125cc_under, "Motorcycle 125cc and unde
```



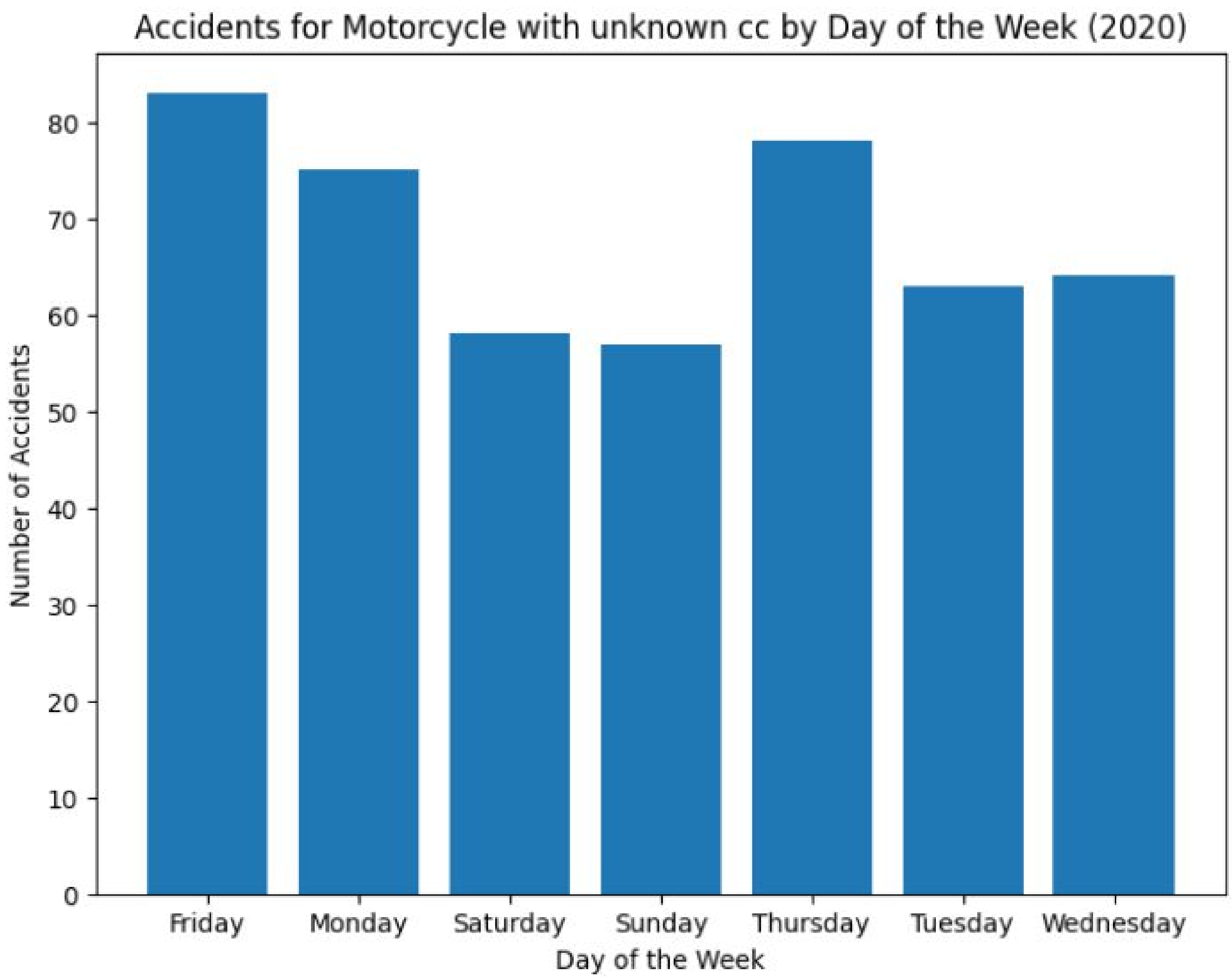
In [93]:
```
accidents_by_day_of_week(motorcycle_125cc_over, "Motorcycle  over 125cc and
```



In [94]:`accidents_by_day_of_week(motorcycle_500cc_over, "Motorcycle over 500cc")`



In [98]:
```
accidents_by_day_of_week(motorcycle_unknowncc, "Motorcycle with unknown cc"
```



In [ ]:

In [ ]:

Merging the Vehicle and Accident data together

In [72]:
```
# Merging the Accident and Vehicle group together to extract the sisgnifica

VEH2020_COMB = pd.merge(ACC2020, VEH2020, on='accident_index', how='inner')
```

In [73]:
```
VEH2020_COMB
```

Out[7

	accid	accid	accid	locat	locat	long	latitu	polic	accid	num	...	journ	sex_	age_	age_	engi	prop	age_	gene	drive	d
0	2020	2020	0102	5213	1751	-0.254	51.46	1	3	1	...	6	2	32	6	1968	2	6	AUD Q5	4	1
1	2020	2020	0102	5293	1762	-0.139	51.47	1	3	1	...	2	1	45	7	1395	1	2	AUD A1	7	1
2	2020	2020	0102	5264	1827	-0.178	51.52	1	3	1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-1
3	2020	2020	0102	5386	1843	-0.001	51.54	1	2	1	...	1	1	44	7	1798	8	8	TOYO PRIU	2	1
4	2020	2020	0102	5293	1812	-0.137	51.51	1	3	1	...	6	1	20	4	2993	2	4	BMW 4 SERII	-1	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1673	2020	2020	9910	3836	8106	-2.271	57.18	99	2	2	...	1	1	57	9	1968	2	2	AUD A5	7	1
1673	2020	2020	9910	3836	8106	-2.271	57.18	99	2	2	...	5	1	38	7	1301	1	2	KTM 1290 SUPE	9	2
1673	2020	2020	9910	2771	6748	-3.968	55.95	99	3	2	...	6	2	68	10	1995	2	1	BMW X3	5	1
1673	2020	2020	9910	2771	6748	-3.968	55.95	99	3	2	...	6	1	76	11	-1	-1	-1	-1	9	1
1673	2020	2020	9910	2404	6819	-4.561	56.00	99	3	1	...	6	1	39	7	999	1	2	FORD FOCU	7	1

Accidents significant hours for Motorcycle 125cc and under,

In [74]:
```
VEH2020_CAT2 = VEH2020_COMB[VEH2020_COMB["vehicle_type"].isin([2, 3])]
VEH2020_CAT2
```

Out[7

	acci	acci	acci	loca	loca	long	latit	polic	acci	num	...	jour	sex_	age_	age_	engi	prop	age_	gene	driv	d
12	2020	2020	0102	5297	1923	-0.12	51.6	1	3	2	...	6	1	37	7	114	1	5	YAM XC11	8	1
32	2020	2020	0102	5314	1745	-0.10	51.4	1	2	2	...	6	1	19	4	-1	-1	-1	-1	5	1
33	2020	2020	0102	5310	1763	-0.11	51.4	1	3	1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
36	2020	2020	0102	5284	1799	-0.15	51.5	1	3	2	...	6	1	20	4	125	1	4	PEU TWE 125	6	1
58	2020	2020	0102	5184	1852	-0.29	51.5	1	3	1	...	6	1	20	4	125	1	0	HON GLR 125	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1670	2020	2020	9910	3406	6740	-2.95	55.9	99	2	2	...	5	1	18	4	-1	-1	-1	-1	2	2
1671	2020	2020	9910	2767	6670	-3.97	55.8	99	2	2	...	2	1	17	4	125	1	2	KEEV RK 125	3	1
1671	2020	2020	9910	2506	6580	-4.38	55.7	99	3	2	...	6	1	47	8	124	1	17	-1	5	1
1672	2020	2020	9910	2681	6566	-4.10	55.7	99	3	2	...	2	1	61	9	124	1	4	-1	3	1
1673	2020	2020	9910	3116	6837	-3.41	56.0	99	2	2	...	1	1	35	6	125	1	3	-1	5	1

In [74]:
```
# Number of vehicles per significant hours
VEH2020_CAT2.groupby("time")["accident_index"].count().sort_values(ascendin
```

Out[74]:
```
time
17:00    90
17:30    81
19:00    80
15:30    79
18:30    77
         ..
08:59     1
09:01     1
02:16     1
09:02     1
09:24     1
Name: accident_index, Length: 1166, dtype: int64
```

In [75]:
```
#Creating a dataframe
VEH2020_CAT2QL = VEH2020_CAT2.groupby("time")["accident_index"].count().res
```

```
# Renaming the datafrmae
VEH2020_CAT2QL.rename(columns = {"accident_index": "number_of_motorcycles"}

VEH2020_CAT2QL
```

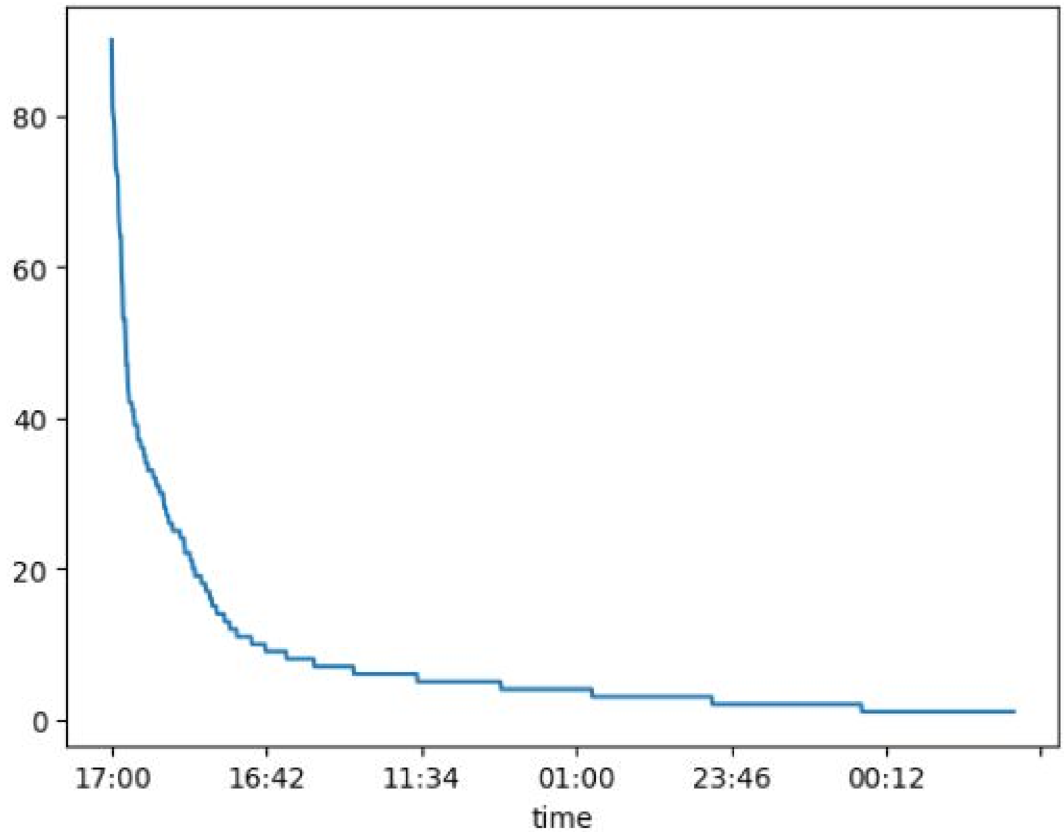
Out[7

	time	number_of_motorcycl
0	00:00	4
1	00:01	6
2	00:02	1
3	00:03	3
4	00:04	3
...	...	...
1161	23:55	1
1162	23:56	2
1163	23:57	2
1164	23:58	1
1165	23:59	2

In [82]:
```
VEH2020_CAT2.groupby("time")["accident_index"].count().sort_values(ascendin
```

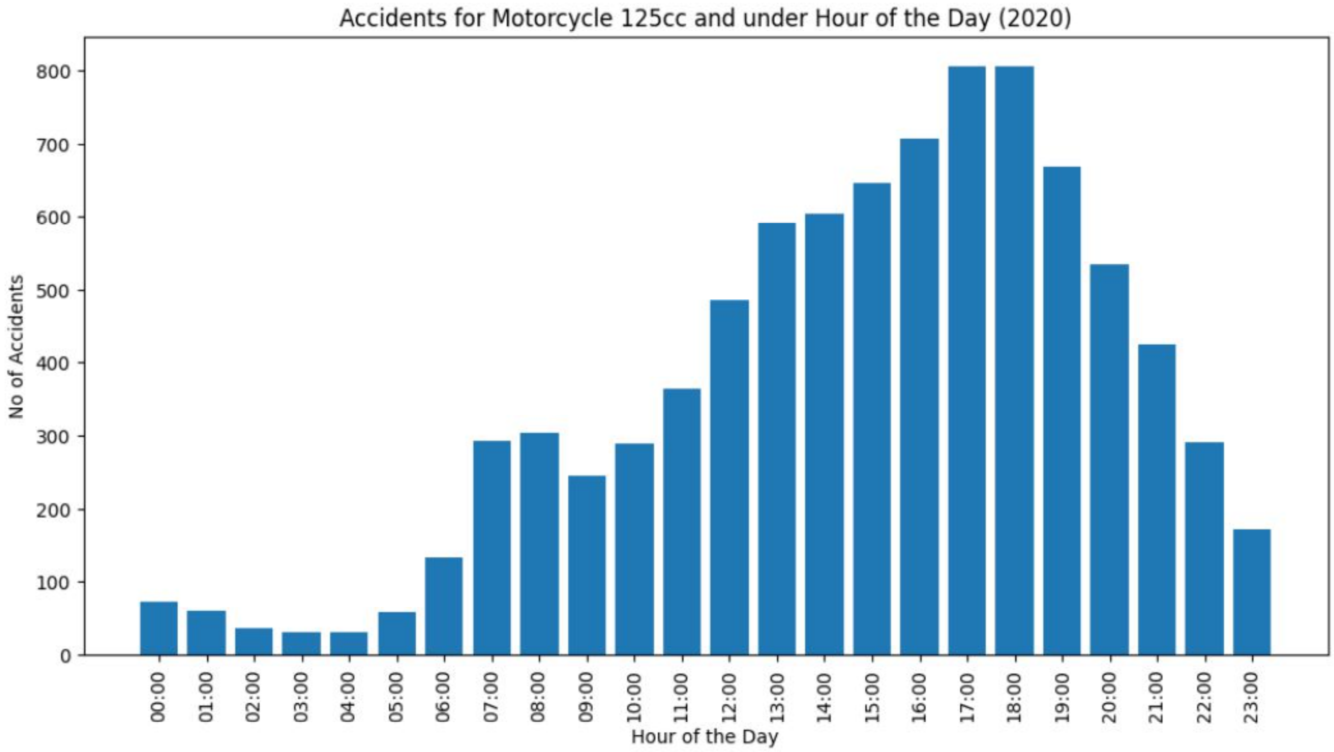
Out[82]:

```
&lt;Axes: xlabel='time'&gt;
```



In [76]:

```
# Plotting for Motorcycle 125cc and under (Vehicle Type 02 or 03)
motorcycle_125cc_under = VEH2020_CAT2[VEH2020_CAT2['vehicle_type'].isin([2,

# hours of the day from the 'time' column
motorcycle_125cc_under.loc[:, 'hour_of_day'] = pd.to_datetime(motorcycle_12
hourly_accidents_125cc_under = motorcycle_125cc_under['hour_of_day'].value_

# Plot the data
plt.figure(figsize=(12, 6))
plt.bar(hourly_accidents_125cc_under.index, hourly_accidents_125cc_under.va
plt.xlabel('Hour of the Day')
plt.ylabel('No of Accidents')
plt.title('Accidents for Motorcycle 125cc and under Hour of the Day (2020)'
plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)
plt.show()
```



The above plot shows the insights of the hourly accidents for Motorcycle 125cc and under

In [ ]:

Accidents significant hours for Motorcycle over 125cc and up to 500cc

In [77]:
```
VEH2020_CAT4 = VEH2020_COMB[VEH2020_COMB["vehicle_type"].isin([4])]
VEH2020_CAT4
```

Out[7

	acci	acci	acci	loca	loca	long	latit	polic	acci	num	...	jour	sex_	age_	age_	engi	prop	age_	gene	driv	d
92	2020	2020	0102	5401	1903	0.02	51.5	1	3	2	...	6	1	39	7	125	1	7	HON SH1	1	1
106	2020	2020	0102	5309	1710	-0.11	51.4	1	3	1	...	6	1	33	6	125	1	3	HON WW	5	1
618	2020	2020	0102	5377	1805	-0.01	51.5	1	3	2	...	2	1	44	7	395	1	10	-1	5	1
722	2020	2020	0102	5257	1750	-0.19	51.4	1	3	3	...	6	1	26	6	499	1	16	-1	3	1
750	2020	2020	0102	5255	1767	-0.19	51.4	1	3	3	...	6	1	18	4	-1	-1	-1	PIAG MP3	3	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1663	2020	2020	9909	3394	7341	-2.98	56.4	99	2	1	...	6	1	21	5	-1	-1	-1	-1	4	1
1667	2020	2020	9910	3615	8624	-2.64	57.6	99	2	3	...	5	1	29	6	249	1	16	-1	3	1
1668	2020	2020	9910	3222	6720	-3.24	55.9	99	2	2	...	5	2	27	6	125	1	1	-1	7	1
1670	2020	2020	9910	3243	6728	-3.21	55.9	99	3	2	...	1	1	38	7	249	1	9	-1	7	1
1673	2020	2020	9910	3180	7452	-3.33	56.5	99	3	2	...	5	1	48	8	250	1	5	-1	2	1

In [78]:
```
VEH2020_CAT4.groupby("time")["accident_index"].count().sort_values(ascendin
```

Out[78]:

```
time
17:00    23
16:00    18
16:30    17
15:30    17
15:00    16
         ..
12:38     1
12:34     1
12:31     1
12:28     1
23:50     1
Name: accident_index, Length: 685, dtype: int64
```

In [79]:
```
#Creating a dataframe
VEH2020_CAT4QL = VEH2020_CAT4.groupby("time")["accident_index"].count().res

# Renaming the datafrmae
VEH2020_CAT4QL.rename(columns = {"accident_index": "number_of_motorcycles"}
```

```
VEH2020_CAT4QL
```

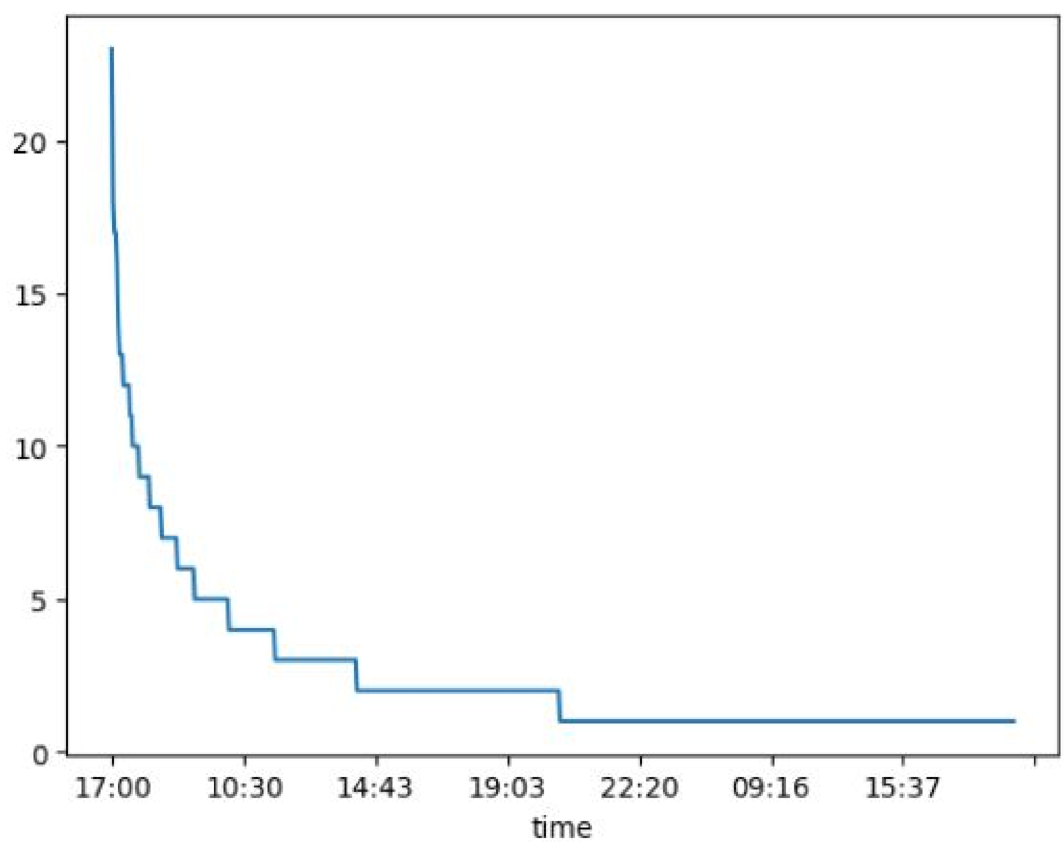
Out[7

		time	number_of_motorcycl
0	00:00	1	
1	00:02	1	
2	00:04	1	
3	00:15	2	
4	00:16	1	
...	...	...	
680	23:41	1	
681	23:42	1	
682	23:43	1	
683	23:45	3	
684	23:50	1	

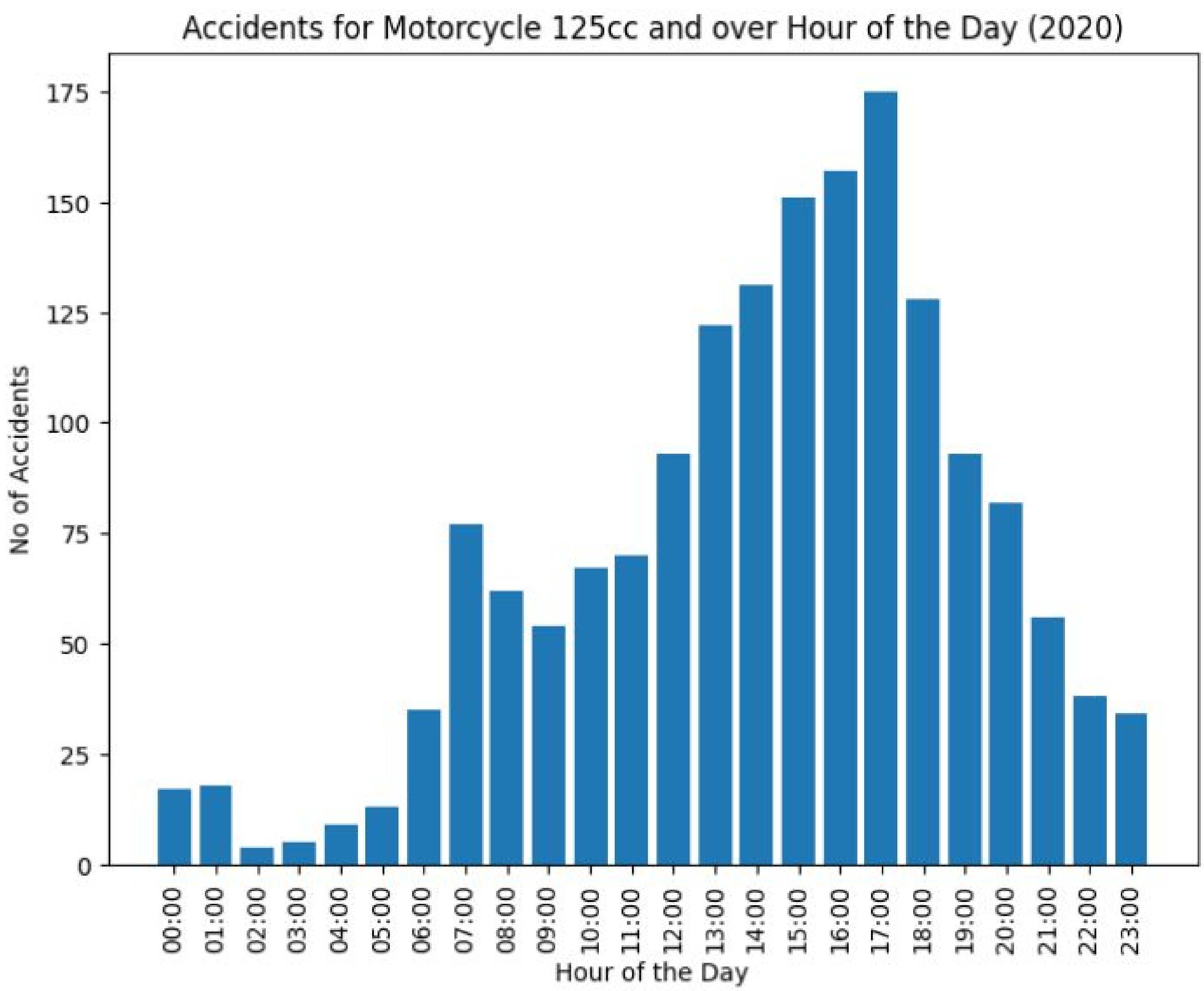
In [83]:
```
VEH2020_CAT4.groupby("time")["accident_index"].count().sort_values(ascendin
```

Out[83]:

```
<Axes: xlabel='time'>
```



In [76]:
```
# Plotting for Motorcycle Motorcycle over 125cc and up to 500cc (Vehicle Ty
motorcycle_125cc_over = VEH2020_CAT4[VEH2020_CAT4['vehicle_type'].isin([4])

# hours of the day from the 'time' column
motorcycle_125cc_over.loc[:, 'hour_of_day'] = pd.to_datetime(motorcycle_125
hourly_accidents_125cc_over = motorcycle_125cc_over['hour_of_day'].value_co

# Plot the data
plt.figure(figsize=(8, 6))
plt.bar(hourly_accidents_125cc_over.index, hourly_accidents_125cc_over.valu
plt.xlabel('Hour of the Day')
plt.ylabel('No of Accidents')
plt.title('Accidents for Motorcycle 125cc and over Hour of the Day (2020)')
plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)
plt.show()
```



In [ ]:

Accidents significant hours for Motorcycle over 500cc

In [84]:
```
VEH2020_CAT5 = VEH2020_COMB[VEH2020_COMB["vehicle_type"].isin([5])]
VEH2020_CAT5
```

Out[8

	accic	accic	accic	locat	locat	long	latitu	polic	accic	num	...	journ	sex_(	age_	age_	engi	prop	age_	gene	drive	d
119	2020	2020	0102	5245	1786	-0.207	51.49	1	2	2	...	6	1	35	6	600	1	8	YAM XJ6	2	1
334	2020	2020	0102	5307	1874	-0.115	51.57	1	3	2	...	6	1	48	8	1200	1	17	HARI - DAVI MOD MISS	2	1
381	2020	2020	0102	5152	1830	-0.339	51.53	1	3	4	...	2	1	40	7	647	1	2	-1	6	1
402	2020	2020	0102	5307	1793	-0.117	51.49	1	3	2	...	6	1	31	6	865	1	15	TRIU BON	4	1
450	2020	2020	0102	5485	1783	0.138	51.48	1	3	2	...	6	1	18	4	847	1	3	YAM MT09	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1667	2020	2020	9910	3615	8624	-2.646	57.64	99	2	3	...	5	1	30	6	599	1	19	-1	6	2
1667	2020	2020	9910	3262	6697	-3.182	55.91	99	2	1	...	5	1	53	8	649	1	3	KAW KLE6	7	1
1668	2020	2020	9910	3385	7318	-2.999	56.47	99	2	2	...	2	1	50	8	600	1	20	KAW MOD MISS	1	1
1671	2020	2020	9910	2761	6638	-3.979	55.85	99	3	2	...	5	1	48	8	649	1	18	-1	8	1
1673	2020	2020	9910	3836	8106	-2.271	57.18	99	2	2	...	5	1	38	7	1301	1	2	KTM 1290 SUPE	9	2

In [85]:
```
VEH2020_CAT5.groupby("time")["accident_index"].count().sort_values(ascendin
```

Out[85]:

```
time
17:30    43
17:00    35
14:00    34
15:00    31
16:00    31
         ..
11:31     1
11:29     1
11:26     1
11:17     1
23:53     1
Name: accident_index, Length: 913, dtype: int64
```

In [78]:
```
#Creating a dataframe
```

```
VEH2020_CAT5QL = VEH2020_CAT5.groupby("time")["accident_index"].count().res

# Renaming the datafrmae
VEH2020_CAT5QL.rename(columns = {"accident_index": "number_of_motorcycles"}

VEH2020_CAT5QL
```

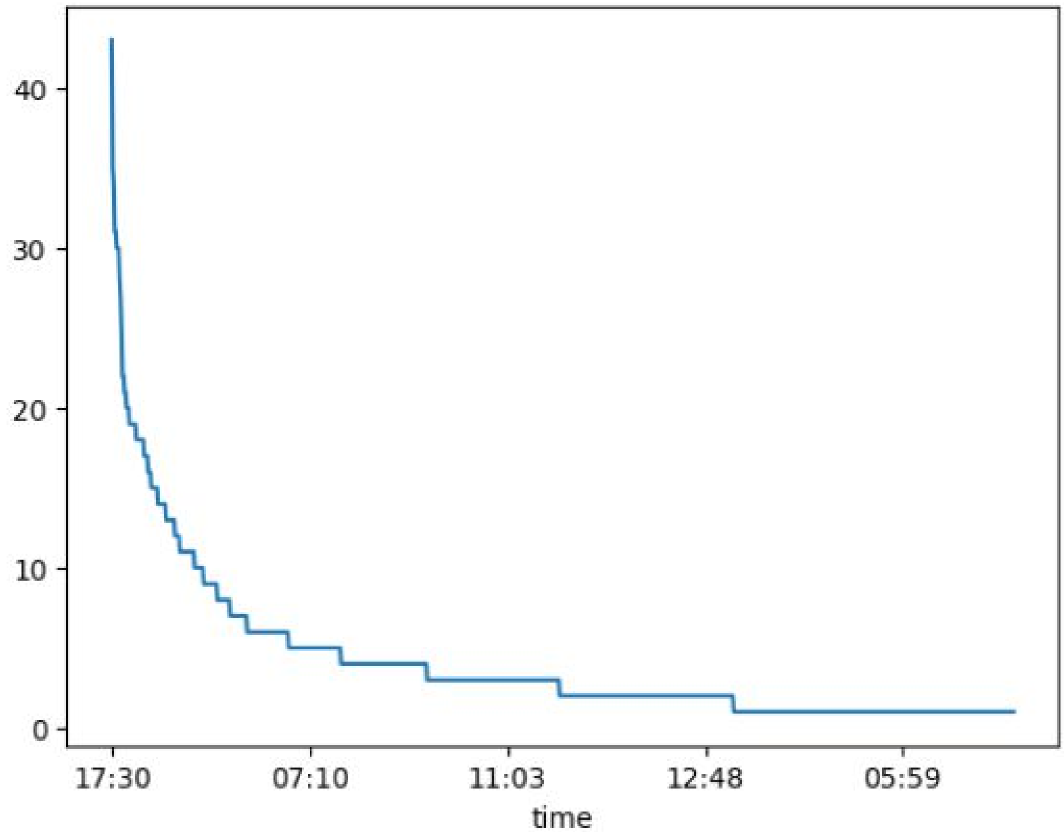
Out[7

	time	number_of_motorcycl
0	00:00	1
1	00:10	1
2	00:14	1
3	00:20	1
4	00:25	1
...	...	...
908	23:40	1
909	23:44	1
910	23:45	1
911	23:50	4
912	23:53	1

In [79]:
```
VEH2020_CAT5.groupby("time")["accident_index"].count().sort_values(ascendin
```

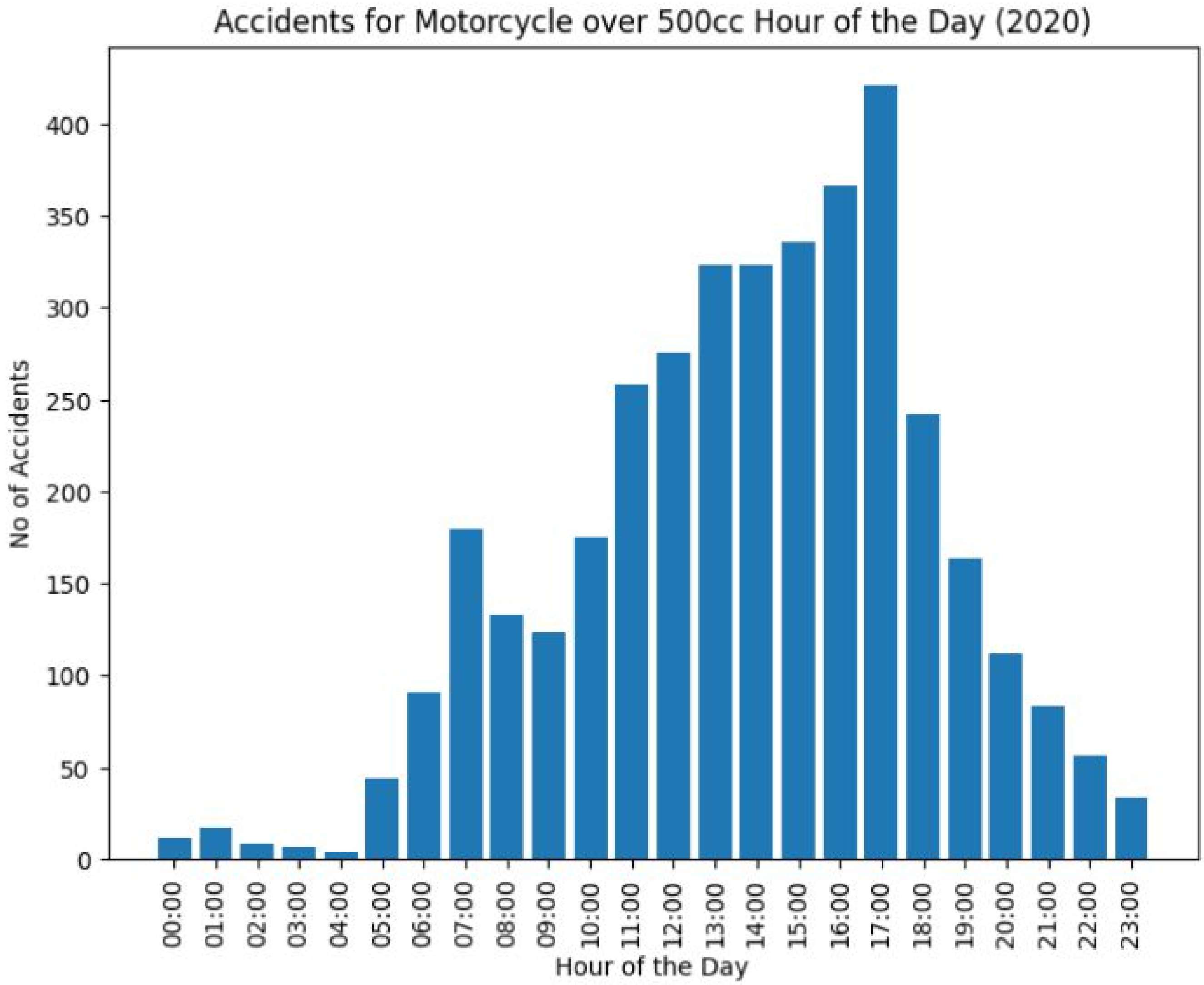
Out[79]:

```
<Axes: xlabel='time'>
```



In [80]:

```
# Plotting for Motorcycle Motorcycle over 500cc (Vehicle Type 5)
motorcycle_500cc_over = VEH2020_CAT5[VEH2020_CAT5['vehicle_type'].isin([5])

# hours of the day from the 'time' column
motorcycle_500cc_over.loc[:, 'hour_of_day'] = pd.to_datetime(motorcycle_500
hourly_accidents_500cc_over = motorcycle_500cc_over['hour_of_day'].value_co

# Plot the data
plt.figure(figsize=(8, 6))
plt.bar(hourly_accidents_500cc_over.index, hourly_accidents_500cc_over.valu
plt.xlabel('Hour of the Day')
plt.ylabel('No of Accidents')
plt.title('Accidents for Motorcycle over 500cc Hour of the Day (2020)')
plt.xticks(range(24), [f'{h:02d}:00' for h in range(24)], rotation=90)
plt.show()
```

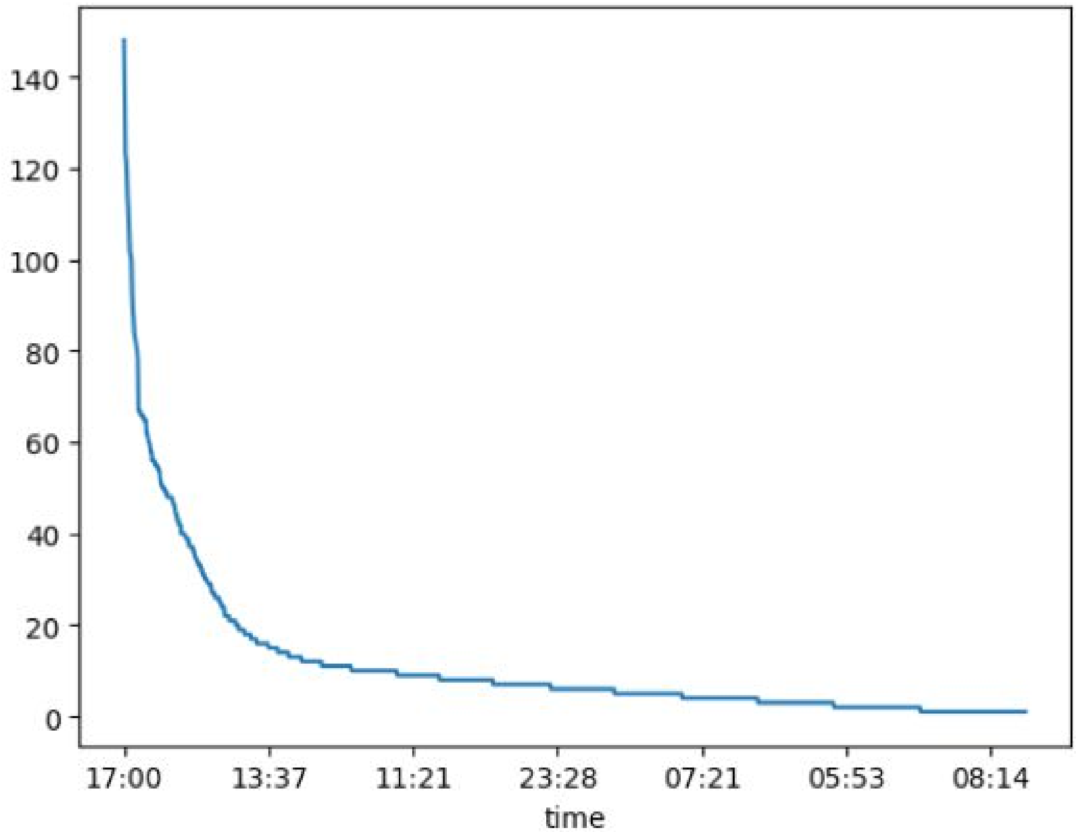


In [ ]:

In [28]:
```
VEH2020_COMB.groupby("time")["accident_index"].count().sort_values(ascendin
```

Out[28]:

<AxesSubplot:xlabel='time'>



In [29]:
```
#Creating a dataframe

VEH2020_COMBQL = VEH2020_COMB.groupby("time")["accident_index"].count().res
```
In [30]:
```
# Renaming the datafrmae

VEH2020_COMBQL.rename(columns = {"accident_index": "number_of_motorcycles"}
```
In [31]:
```
VEH2020_COMBQL.head(10)
```

Out[3

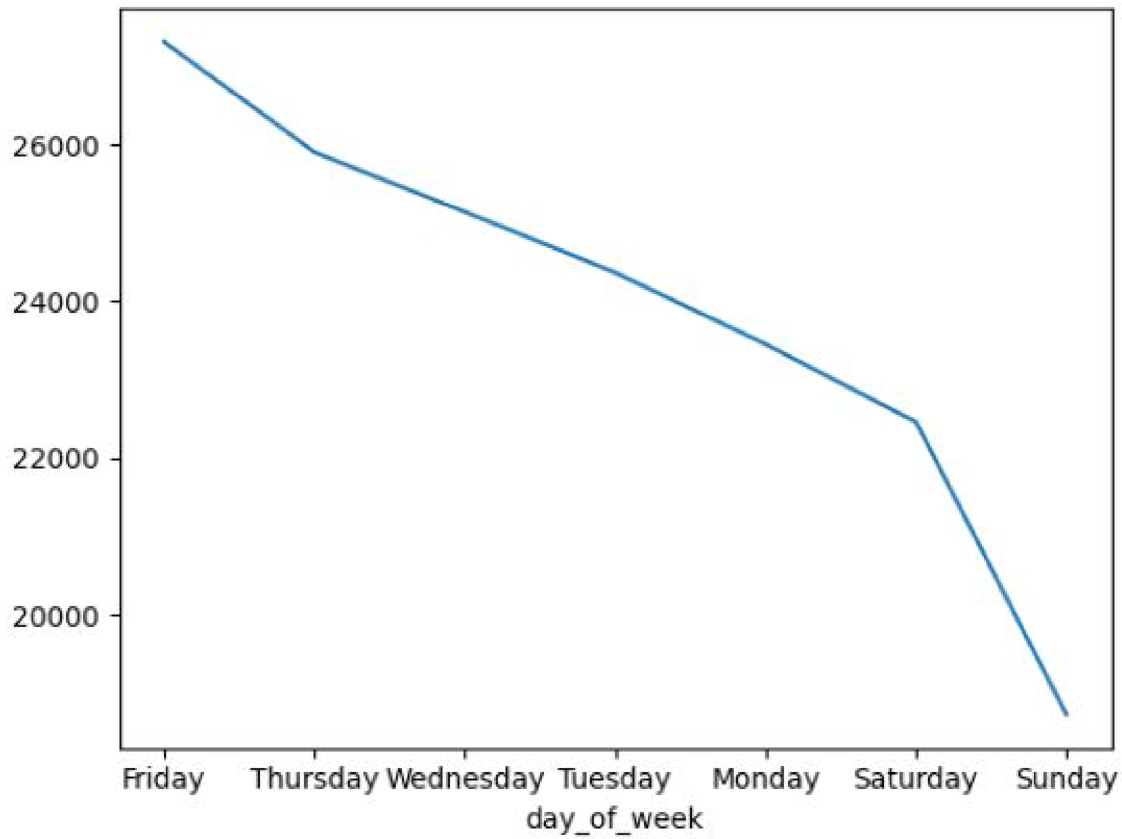
		time	number_of_motorcycl
0	00:00	6	
1	00:01	6	
2	00:02	2	
3	00:03	3	
4	00:04	4	
5	00:05	3	
6	00:07	1	
7	00:10	6	
8	00:12	1	

significant days of the week, on which accidents occur by the vehicle group

In [81]:
```
VEH2020_COMB["date"] = pd.to_datetime(VEH2020_COMB["date"], dayfirst = True
```
In [82]:
```
VEH2020_COMB["day_of_week"]= VEH2020_COMB["date"].dt.day_name()
```
In [83]:
```
VEH2020_COMB["day_of_week"]
```
Out[83]:
```
0           Tuesday
1            Monday
2         Wednesday
3         Wednesday
4         Wednesday
            ...
167370    Wednesday
167371    Wednesday
167372      Tuesday
167373      Tuesday
167374      Tuesday
Name: day_of_week, Length: 167375, dtype: object
```
In [84]:
```
VEH2020_COMB.groupby("day_of_week")["accident_index"].count().sort_values(a
```
Out[84]:
```
day_of_week
Friday       27314
Thursday     25905
Wednesday    25145
Tuesday      24368
Monday       23451
Saturday     22464
Sunday       18728
Name: accident_index, dtype: int64
```
In [85]:
```
# Showing significant days of the week on which highest accident occur

VEH2020_COMB.groupby("day_of_week")["accident_index"].count().sort_values(a
```

Out[85]:

```
<Axes: xlabel='day_of_week'>
```



In [ ]:

3. For pedestrians involved in accidents, are there significant hours of the day, and days of the week, on which they are more likely to be involved?

Using data from the Casualty Class

In [100]:
```
#Showing the unique Casualty class from the CAS2020 data

CAS2020.casualty_type.unique()
```

Out[100]:
```
array([ 0,  9,  3,  8,  2, 11,  1, 19,  4,  5, 20, 90, 21, 23, 98, 22, 10,
       97, 17, 16, 18], dtype=int64)
```

Focusing on Pedestrian = 3 as shown in the Accident Statistics form

In [101]:
```
CAS2020_PED = CAS2020[CAS2020["casualty_class"] == 3]
```

In [103]:
```
CAS2020_PED
```

Out[10

	casu	accid	accid	accid	vehic	casu	casu	sex_c	age_c	age_b	casu	pede	pede	car_p	bus_c	pede	casu	casu	ca
4847	4847	2020	2020	0102	1	1	3	1	31	6	3	9	5	0	0	0	0	1	4
4847	4847	2020	2020	0102	1	1	3	2	2	1	3	1	1	0	0	0	0	1	2
4847	4847	2020	2020	0102	1	2	3	2	4	1	3	1	1	0	0	0	0	1	2
4847	4847	2020	2020	0102	1	1	3	1	23	5	3	5	9	0	0	0	0	1	3
4847	4847	2020	2020	0102	1	1	3	1	47	8	2	4	1	0	0	0	0	1	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6003	6003	2020	2020	9910	1	2	3	2	54	8	2	8	1	0	0	0	0	1	4
6003	6003	2020	2020	9910	1	1	3	2	58	9	3	5	1	0	0	0	0	1	4
6003	6003	2020	2020	9910	1	1	3	1	69	10	3	6	9	0	0	0	0	3	7
6003	6003	2020	2020	9910	1	1	3	2	63	9	3	10	1	0	0	0	0	1	10
6003	6003	2020	2020	9910	1	1	3	1	48	8	3	9	9	0	0	0	0	1	1

In [104]:
```
# Merging the Accident data and CAS2020_PED

CAS2020_PED_COMB = pd.merge(ACC2020, CAS2020_PED, on = "accident_index")
```

In [105]:
```
# Showing rows

CAS2020_PED_COMB
```

Out[10

	accic	accic	accic	locat	locat	long	latitu	polic	accic	num	...	age_	casu	pede	pede	car_p	bus_	pede	casu	casu	ca
0	2020	2020	0102	5213	1751	-0.254	51.46	1	3	1	...	6	3	9	5	0	0	0	0	1	4
1	2020	2020	0102	5293	1762	-0.139	51.47	1	3	1	...	1	3	1	1	0	0	0	0	1	2
2	2020	2020	0102	5293	1762	-0.139	51.47	1	3	1	...	1	3	1	1	0	0	0	0	1	2
3	2020	2020	0102	5264	1827	-0.178	51.52	1	3	1	...	5	3	5	9	0	0	0	0	1	3
4	2020	2020	0102	5386	1843	-0.001	51.54	1	2	1	...	8	2	4	1	0	0	0	0	1	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1474	2020	2020	9910	3417	7336	-2.946	56.49	99	2	1	...	8	2	8	1	0	0	0	0	1	4
1474	2020	2020	9910	3429	7311	-2.927	56.46	99	3	1	...	9	3	5	1	0	0	0	0	1	4
1474	2020	2020	9910	2862	7170	-3.841	56.33	99	3	1	...	10	3	6	9	0	0	0	0	3	7
1474	2020	2020	9910	2579	6588	-4.267	55.80	99	3	1	...	9	3	10	1	0	0	0	0	1	10
1474	2020	2020	9910	2404	6819	-4.561	56.00	99	3	1	...	8	3	9	9	0	0	0	0	1	1

Significant hours of the day on which Pedestrians are more likely to be involved

In [106]:
```
# significant hours of the day on which accident occur

CAS2020_PED_COMB.groupby("time")["accident_index"].count().sort_values(asc
```

Out[106]:

```
time
15:30    188
15:00    164
16:00    153
18:00    152
17:00    150
        ...
03:46      1
02:11      1
19:34      1
06:53      1
01:43      1
Name: accident_index, Length: 1264, dtype: int64
```

In [44]:
```
#Creating a dataframe
```

```
CAS2020_PED_COMBQL = CAS2020_PED_COMB.groupby("time")["accident_index"].cou
```

In [45]:
```
# Renaming the datafrmae

CAS2020_PED_COMBQL.rename(columns = {"accident_index": "number_of_pedestria
```

In [46]:
```
CAS2020_PED_COMBQL.head(10)
```

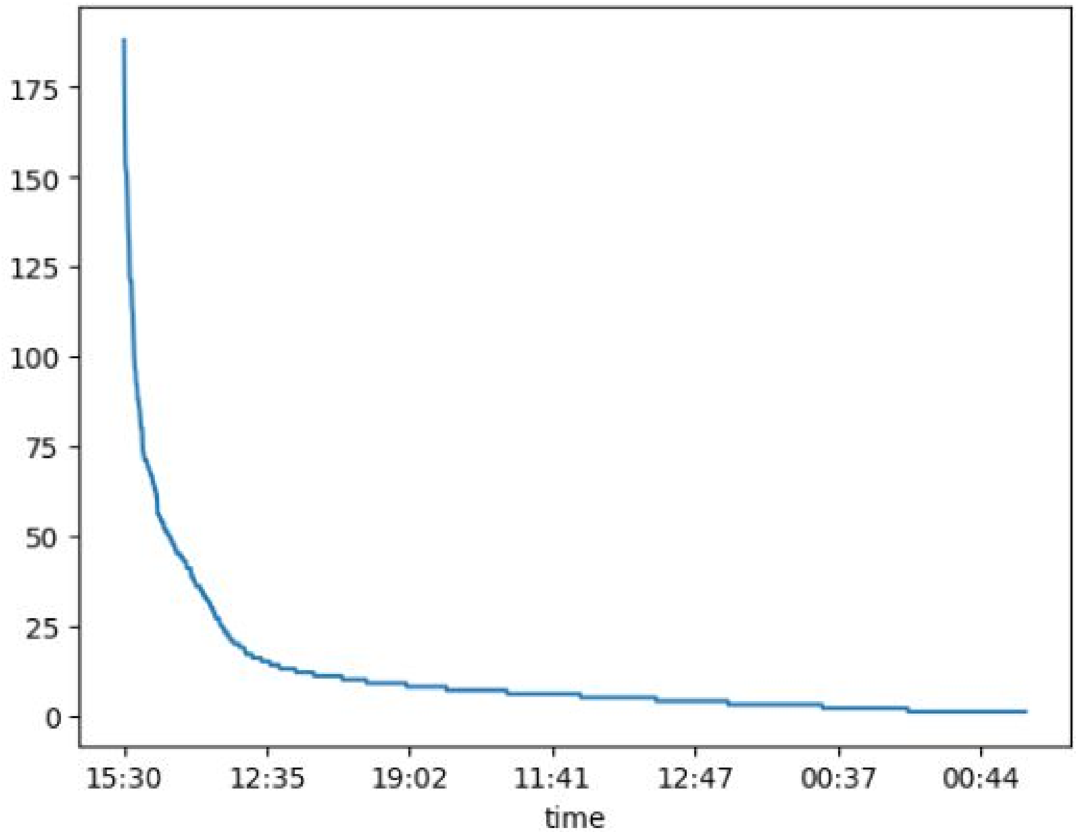
Out[4

	time	number_of_pedestria
0	00:00	9
1	00:01	14
2	00:02	3
3	00:03	1
4	00:04	2
5	00:05	7
6	00:06	2
7	00:07	1
8	00:08	1

In [47]:
```
# Showing the plot with the above dataframe

CAS2020_PED_COMB.groupby("time")["accident_index"].count().sort_values(asce
```

Out[47]:

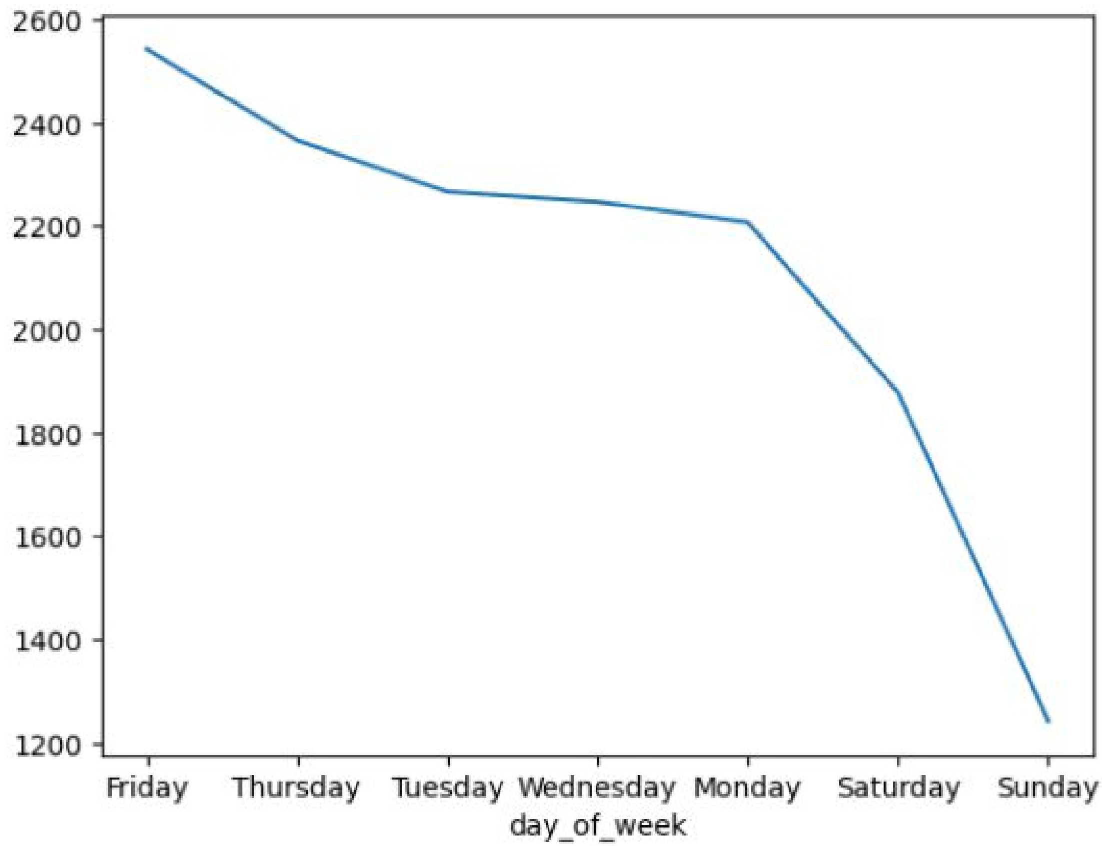
```
<AxesSubplot:xlabel='time'>
```



Significant days of the week on which Pedestrians are more likely to be involved

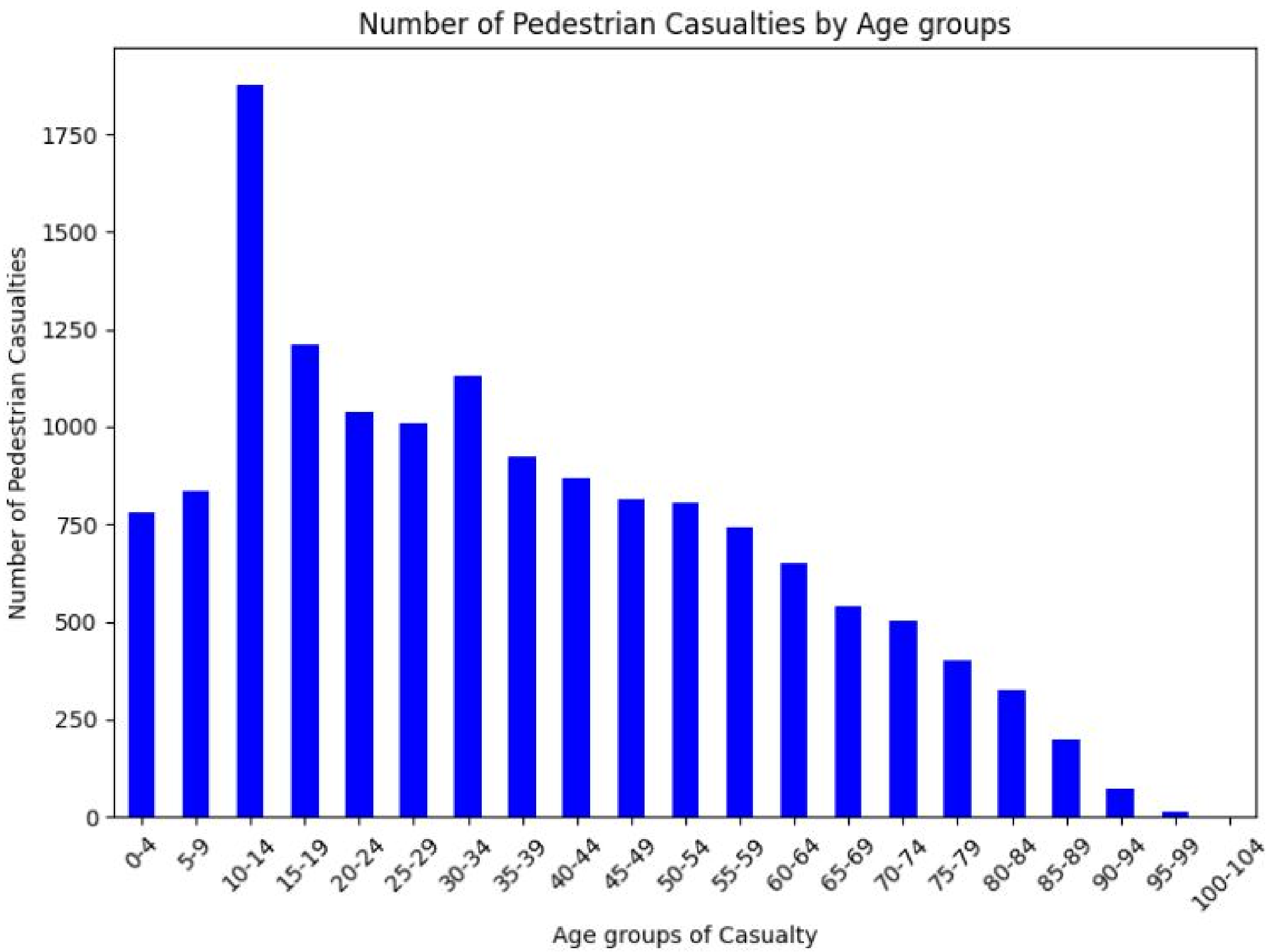
In [48]:
```
CAS2020_PED_COMB["date"] = pd.to_datetime(CAS2020_PED_COMB["date"], dayfirs
```
In [49]:
```
CAS2020_PED_COMB["day_of_week"]= CAS2020_PED_COMB["date"].dt.day_name()
```
In [50]:
```
CAS2020_PED_COMB["day_of_week"]
```
Out[50]:

```
0          Tuesday
1           Monday
2           Monday
3        Wednesday
4        Wednesday
           ...
14745      Tuesday
14746       Monday
14747      Tuesday
14748       Friday
14749      Tuesday
Name: day_of_week, Length: 14750, dtype: object
```
In [51]:
```
CAS2020_PED_COMB.groupby("day_of_week")["accident_index"].count().sort_valu
```



Number of Pedestrian Casualties by Age groups

In [109]:
```
# Define the age ranges
age_grp = range(0, 101, 5)

# Create labels for age ranges
age_labels = [f"{start}-{start+4}" for start in age_grp]

# Group the data by age grp and count the number of casualties in each age
age_nums = CAS2020_PED_COMB['age_of_casualty'].apply(lambda age: next((lab
age_nums = age_nums.value_counts().reindex(age_labels, fill_value=0)

# Create a bar plot
plt.figure(figsize=(8, 6))
age_nums.plot(kind='bar', color='blue')
plt.xlabel('Age groups of Casualty')
plt.ylabel('Number of Pedestrian Casualties')
plt.title('Number of Pedestrian Casualties by Age groups')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



4. Using the apriori algorithm, explore the impact of selected variables on accident severity

Selecting the Features for the Apriori Algorithm from the Accident Data

- ACCIDENT: accident_index, accident_severity, number_of_vehicles, number_of_casualties, road_type, speed_limit, light_conditions, weather_conditions, road_surface_conditions, lsoa_of _accident_location

- VEHICLE: accident_index, vehicle_type, sex_of_driver, age_of_driver,engine_capacity_cc, age_of_vehicle

- CASUALTY: acident_index, casualty_class, sex_of_casualty, pedestrian_location

In [52]:
```
ACC2020_FEAT = ["accident_index", "accident_severity", "number_of_vehicles"
VEH2020_FEAT = ["accident_index", "vehicle_type", "sex_of_driver", "age_of_
CAS2020_FEAT = ["accident_index", "casualty_class", "sex_of_casualty", "ped
```

In [53]:
```
#Subsetting the data based on the above selected features
```

```
ACC2020_SUB = ACC2020[ACC2020_FEAT]
VEH2020_SUB = VEH2020[VEH2020_FEAT]
CAS2020_SUB = CAS2020[CAS2020_FEAT]
```

In [54]:
```
# checking the accident features

ACC2020_SUB.head()
```

Out[5

	accident_i	accident_s	number_o	number_o	road_type	speed_lim	light_conc	weather_c	road_surfa	lsoa_of
370153	202001021	3	1	1	6	20	1	9	9	E01004
370154	202001022	3	1	2	6	20	1	1	1	E01003
370155	202001022	3	1	1	6	30	4	1	2	E01004
370156	202001022	2	1	1	6	30	4	1	1	E01003

In [55]:
```
# checking the vehicle features

VEH2020_SUB.head()
```

Out[5

	accident_index	vehicle_type	sex_of_driver	age_of_driver	engine_capacity	age_of_vehic
681716	2020010219808	9	2	32	1968	6
681717	2020010220496	9	1	45	1395	2
681718	2020010228005	9	3	-1	-1	-1
681719	2020010228006	8	1	44	1798	8

In [56]:
```
# checking the casualty features

CAS2020_SUB.head()
```

Out[5

	accident_index	casualty_class	sex_of_casualty	pedestrian_locatic
484748	2020010219808	3	1	9
484749	2020010220496	3	2	1
484750	2020010220496	3	2	1
484751	2020010228005	3	1	5

Merging the whole datasets for the Apriori Algorithm. The ACCIDENT and VEHICLE datasets will be merged and the result will now be merged with CASUALTY data

In [57]:
```
# merging ACC2020_SUB + VEH2020_SUB
ACC_VEH_2020SUB = ACC2020_SUB.merge(VEH2020_SUB, on = "accident_index")

# Merging the result + CAS2020_SUB
ACC_VEH_CAS_2020SUB = ACC_VEH_2020SUB.merge(CAS2020_SUB, on = "accident_ind
```

In [58]:
```
ACC_VEH_CAS_2020SUB.head()
```

Out[5

	accid	accid	numb	numb	road_	speec	light_	weatl	road_	lsoa_	vehic	sex_o	age_c	engin	age_c	casua	sex_o	pe
0	2020(	3	1	1	6	20	1	9	9	E010(	9	2	32	1968	6	3	1	9
1	2020(	3	1	2	6	20	1	1	1	E010(	9	1	45	1395	2	3	2	1
2	2020(	3	1	2	6	20	1	1	1	E010(	9	1	45	1395	2	3	2	1
3	2020(	3	1	1	6	30	4	1	2	E010(	9	3	-1	-1	-1	3	1	5

In [59]:
```
#Showing the number of rows & columns

ACC_VEH_CAS_2020SUB.shape
```
Out[59]:
```
(220435, 18)
```

One-hot Encoding: Removing and dropping the continuous and categorical values for one-hot encoding to complete the Apriori Algorithm

In [60]:
```
data = ACC_VEH_CAS_2020SUB.drop(["accident_index", "speed_limit", "number_o
                                 "age_of_driver", "engine_capacity_cc", "age_of
```
In [61]:
```
data.head()
```
Out[6

	accident_s	road_type	light_conc	weather_c	road_surfa	vehicle_ty	sex_of_dri	casualty_c	sex_of_cas	pedest
0	3	6	1	9	9	9	2	3	1	9
1	3	6	1	1	1	9	1	3	2	1
2	3	6	1	1	1	9	1	3	2	1
3	3	6	4	1	2	9	3	3	1	5

In [62]:
```
# Coding the data

def encode_features(data):
    if data <= 1:
           return 0
    else:
        return 1
```
In [63]:
```
# Applying the function to all the columns

data_encoded = data.applymap(encode_features)
```
In [64]:
```
data_encoded.head()
```

Out[6

	accident_s	road_type	light_conc	weather_c	road_surfa	vehicle_ty	sex_of_dri	casualty_c	sex_of_cas	pedest
0	1	1	0	1	1	1	1	1	0	1
1	1	1	0	0	0	1	0	1	1	0
2	1	1	0	0	0	1	0	1	1	0
3	1	1	1	0	1	1	1	1	0	1

Apriori Algorithm

In [ ]:
```
!pip install mlxtend
import mlxtend
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

In [65]:
```
frq_features = apriori(data_encoded, min_support = 0.05, use_colnames=True)
```

```
C:\Users\757538\AppData\Roaming\Python\Python310\site-packages\mlxtend\frequent
_patterns\fpcommon.py:110: DeprecationWarning: DataFrames with non-bool types r
esult in worse computationalperformance and their support might be discontinue
d in the future.Please use a DataFrame with bool type
  warnings.warn(
```

In [66]:
```
# showing the Support
frq_features
```

Out[6

	support	itemse
0	0.980806	(accident_severity)
1	0.943661	(road_type)
2	0.289396	(light_conditions)
3	0.219108	(weather_conditions)
4	0.311584	(road_surface_conditions)
...	...	...
229	0.050196	(casualty_class, road_type, sex_of_drive sex...
230	0.070760	(road_type, road_surface_conditions, weather_c...
231	0.059056	(road_type, road_surface_conditions, weather_c...
232	0.061456	(road_type, road_surface_conditions, weather_c...
233	0.059718	(road_type, road_surface_conditions, sex_of_dr...

Showing the Association and Setting Metrics

In [67]:
```
pd.set_option("max_colwidth", None)
```
In [68]:
```
rules = association_rules(frq_features, metric = "confidence", min_threshol
```
In [69]:
```
rules
```
Out[6

	anteceder	consequer	anteceder support	conseque support	support	confidenc	lift	leverage	convictior	zhangs
0	(road_type	(accident_	0.943661	0.980806	0.924685	0.979891	0.999067	-0.000864	0.954490	-0.01630
1	(accident_	(road_type	0.980806	0.943661	0.924685	0.942781	0.999067	-0.000864	0.984612	-0.04640
2	(light_conc	(accident_	0.289396	0.980806	0.282129	0.974888	0.993966	-0.001713	0.764316	-0.00847
3	(weather_c	(accident_	0.219108	0.980806	0.215161	0.981987	1.001204	0.000259	1.065568	0.00154
4	(road_surf	(accident_	0.311584	0.980806	0.304466	0.977156	0.996279	-0.001137	0.840224	-0.00539
...	...	...	...	...	...	...	...	...	...	...
607	(road_surf sex_of_driv sex_of_cas vehicle_typ accident_s	(road_type	0.063211	0.943661	0.059718	0.944739	1.001142	0.000068	1.019509	0.0012
608	(sex_of_ca road_type, road_surfa sex_of_driv	(vehicle_ty accident_s	0.063461	0.902366	0.059718	0.941025	1.042842	0.002453	1.655521	0.04386
609	(sex_of_ca road_surfa vehicle_typ sex_of_driv	(road_type accident_s	0.064064	0.924685	0.059718	0.932163	1.008086	0.000479	1.110224	0.00857
610	(sex_of_ca road_surfa accident_s sex_of_driv	(road_type vehicle_typ	0.066423	0.872058	0.059718	0.899058	1.030961	0.001793	1.267479	0.03216
611	(sex_of_ca road_surfa sex_of_driv	(accident_ road_type, vehicle_typ	0.067281	0.853821	0.059718	0.887600	1.039563	0.002273	1.300530	0.04080

In [70]:
```
# Show 10 rows

rules.sample(10)
```

Out[7

	anteceder	conseque	anteceder support	conseque support	support	confidenc	lift	leverage	convictior	zhangs
205	(road_surf vehicle_typ	(road_type accident_s	0.294858	0.924685	0.273650	0.928074	1.003664	0.000999	1.047108	0.00517
6	(accident_	(vehicle_ty	0.980806	0.920802	0.902366	0.920025	0.999156	-0.000763	0.990279	-0.04216
504	(casualty_c sex_of_driv vehicle_typ	(road_type accident_s	0.099122	0.924685	0.093565	0.943936	1.020819	0.001908	1.343369	0.02263
253	(casualty_c pedestrian road_type)	(accident_	0.060421	0.980806	0.058530	0.968691	0.987648	-0.000732	0.613053	-0.01313
98	(road_type light_cond	(vehicle_ty	0.272865	0.920802	0.258793	0.948428	1.030002	0.007538	1.535679	0.04005
115	(road_surf sex_of_driv	(road_type	0.112496	0.943661	0.106535	0.947012	1.003550	0.000377	1.063230	0.00398
600	(sex_of_cas road_type, road_surfa weather_c	(vehicle_ty accident_s	0.064164	0.902366	0.061456	0.957791	1.061422	0.003556	2.313130	0.06183
70	(casualty_c weather_c	(accident_	0.062154	0.980806	0.060539	0.974016	0.993077	-0.000422	0.738694	-0.00737
154	(sex_of_cas casualty_cl	(vehicle_ty	0.143543	0.920802	0.142387	0.991941	1.077258	0.010212	9.827379	0.08373
461	(sex_of_dri weather_c	(accident_ road_type,	0.082346	0.853821	0.074267	0.901884	1.056292	0.003958	1.489862	0.05807

Looking at the confidence 0.97 which is greater than 0.7 , shows that the weather conditions is associated with the accident severity

5. Identify accidents in our region: Kingston upon Hull, Humberside, and the East Riding of Yorkshire etc. You can do this by filtering on LSOA, or police region or another method if you can find one. Run clustering on this data. What do these clusters reveal about the distribution of accidents in our region?

In [69]:
```
humberside_lsoa_code_start = 'E01012756'
humberside_lsoa_code_end = 'E01013334'
humberside_ACC2020 = ACC2020[
    (ACC2020['lsoa_of_accident_location'] >= humberside_lsoa_code_start) &
    (ACC2020['lsoa_of_accident_location'] <= humberside_lsoa_code_end)]
```

In [71]:
```
# Check for missing values in the humberside accident DataFrame
print("Missing Values in Accident Dataset:")
print()
print(humberside_ACC2020.isna().sum())
```

```
Missing Values in Accident Dataset:

accident_index                                  0
accident_year                                   0
accident_reference                              0
location_easting_osgr                           0
location_northing_osgr                          0
longitude                                       0
latitude                                        0
police_force                                    0
accident_severity                               0
number_of_vehicles                              0
number_of_casualties                            0
date                                            0
day_of_week                                     0
time                                            0
local_authority_district                        0
local_authority_ons_district                    0
local_authority_highway                         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
did_police_officer_attend_scene_of_accident     0
trunk_road_flag                                 0
lsoa_of_accident_location                       0
hour_of_day                                     0
dtype: int64
```

In [72]:
```
from sklearn.cluster import KMeans
```

In [ ]:

In [73]:
```
Geo = humberside_ACC2020[["longitude", "latitude"]]
Geo
```

Out[7

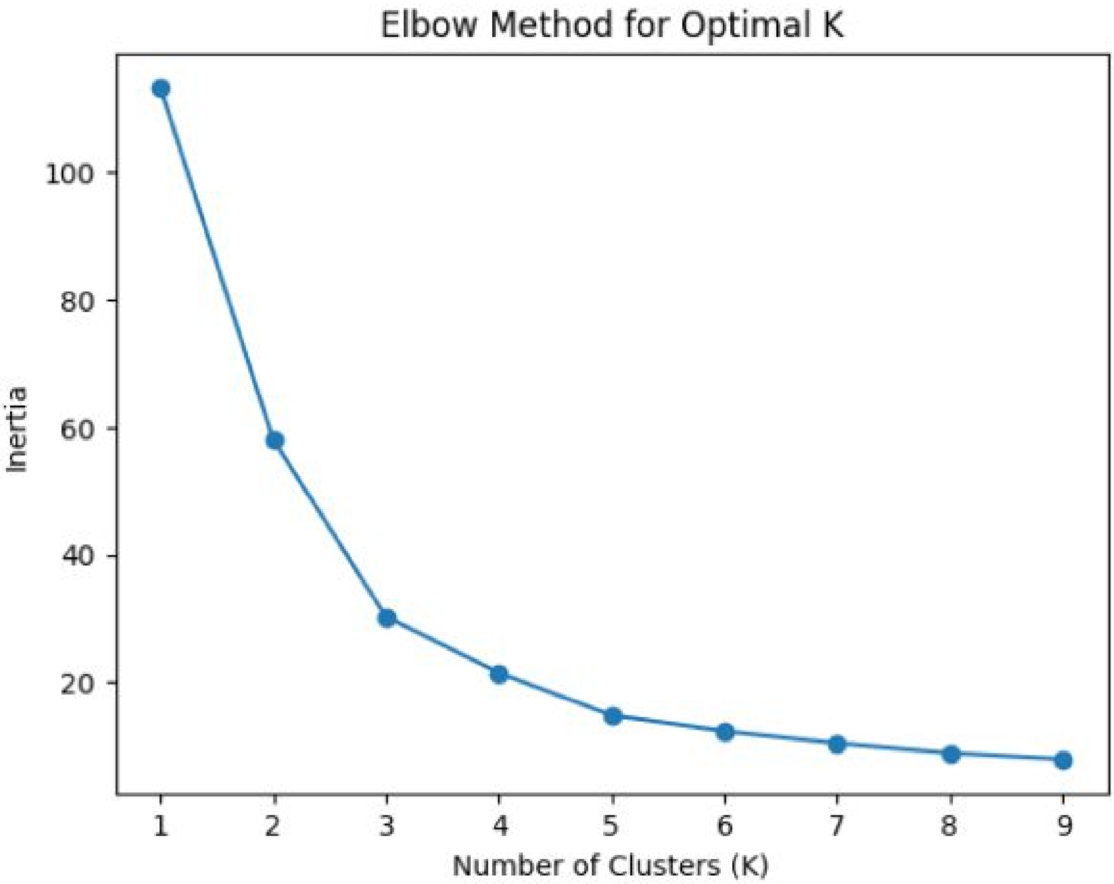
	longitude	latitu
407904	-0.393424	53.744936
407905	-0.528743	53.512895
407906	-0.324858	53.791630
407907	-0.095008	53.574501
407908	-0.327733	53.767805
...	...	...
409607	-0.651104	53.566753
409608	-0.424674	53.839482
409609	-0.308880	53.782750
409610	-0.703181	53.569801
409611	-0.342063	53.742609

In [74]:
```
Geo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1663 entries, 407904 to 409611
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   longitude  1663 non-null   float64
 1   latitude   1663 non-null   float64
dtypes: float64(2)
memory usage: 39.0 KB
```

Using Elbow method to show how good the number of clusters are

In [77]:
```
# Try different values of K
k_values = range(1, 10)
inertias = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, n_init=10)
    kmeans.fit(Geo)
    inertias.append(kmeans.inertia_)

# Plot the inertia values
plt.plot(k_values, inertias, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()
```



In [90]:
```
Kmeans = KMeans(n_clusters = 2,random_state=0, n_init=10).fit(Geo)

Geo
```

Out[9

	longitude	latitud
407904	-0.393424	53.744936
407905	-0.528743	53.512895
407906	-0.324858	53.791630
407907	-0.095008	53.574501
407908	-0.327733	53.767805
...	...	...
409607	-0.651104	53.566753
409608	-0.424674	53.839482
409609	-0.308880	53.782750
409610	-0.703181	53.569801
409611	-0.342063	53.742609

In [91]:
```
labels =Kmeans.predict(Geo)
labels
```

Out[91]:

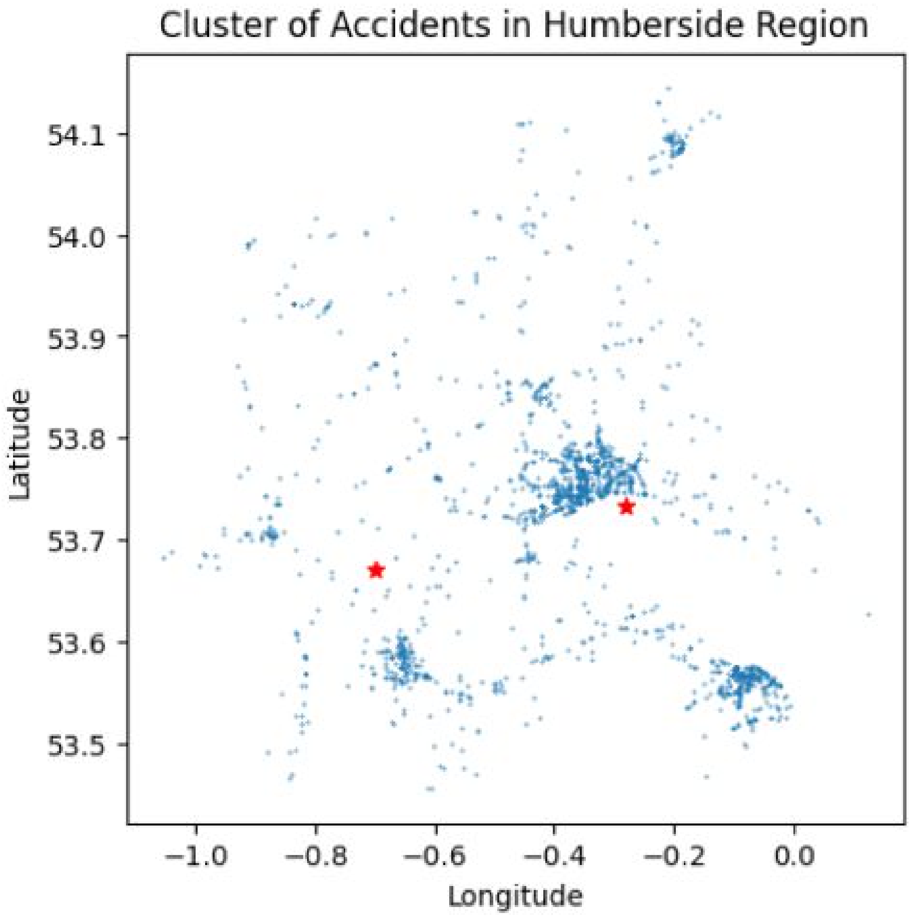
```
array([0, 1, 0, ..., 0, 1, 0])
```

In [92]:
```
centroids = Kmeans.cluster_centers_
centroids
```

Out[92]:

```
array([[-0.2805003 , 53.73377844],
       [-0.70104377, 53.67175429]])
```

In [93]:
```
fig = plt.figure(figsize=(5,5))
plt.scatter(Geo['longitude'], Geo['latitude'], s = 0.5, marker = '.' )
plt.scatter(centroids[:,0], centroids[:,1], color='r', marker ="*")
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Cluster of Accidents in Humberside Region')
plt.show()
```



In [94]:`Kmeans.inertia_`

Out[94]:

```
58.095291259563695
```

n_clusters of 4

In [95]:
```
Kmeans = KMeans(n_clusters = 4,random_state=0, n_init=10).fit(Geo)
Geo
```

Out[9

	longitude	latitu
407904	-0.393424	53.744936
407905	-0.528743	53.512895
407906	-0.324858	53.791630
407907	-0.095008	53.574501
407908	-0.327733	53.767805
...	...	...
409607	-0.651104	53.566753
409608	-0.424674	53.839482
409609	-0.308880	53.782750
409610	-0.703181	53.569801
409611	-0.342063	53.742609

In [96]:
```
Kmeans.inertia_
```

Out[96]:

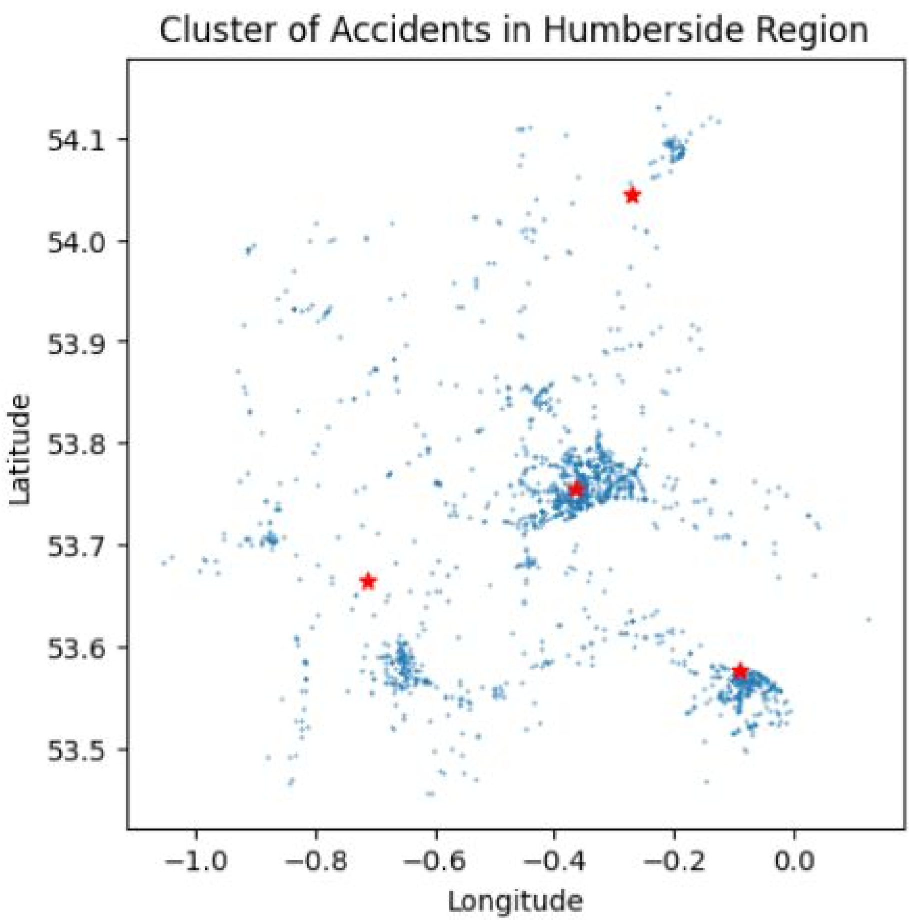
```
21.518076924980857
```

In [97]:
```
centroids = Kmeans.cluster_centers_
centroids
```

Out[97]:

```
array([[-0.36418796, 53.756367  ],
       [-0.09147249, 53.57621063],
       [-0.71434247, 53.66534987],
       [-0.2715766 , 54.0433033 ]])
```

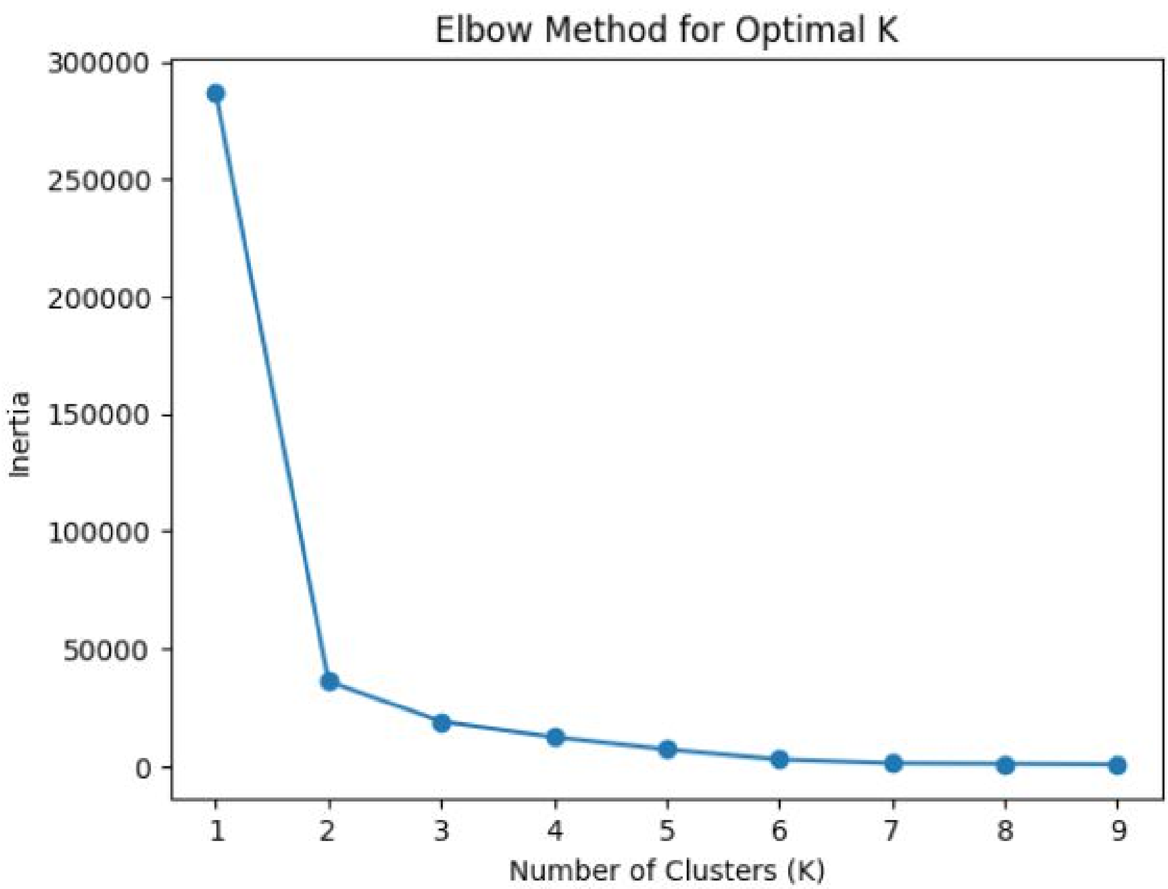
In [98]:
```
fig = plt.figure(figsize=(5,5))
plt.scatter(Geo['longitude'], Geo['latitude'], s = 0.5, marker = '.' )
plt.scatter(centroids[:,0], centroids[:,1], color='r', marker ="*")
plt.title('Cluster of Accidents in Humberside Region')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



Using k-means with 2 clusters with speed_limit and weather_conditions.

In [99]:
```
Geo2 = humberside_ACC2020[["speed_limit", "weather_conditions"]]
```

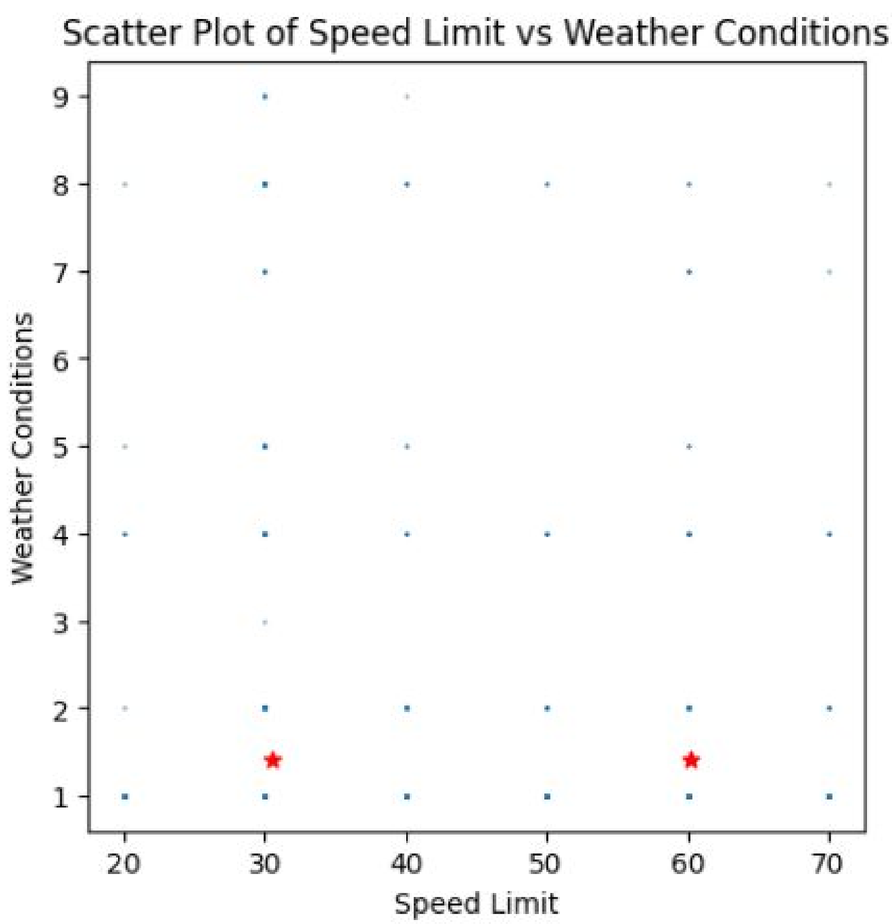
In [100]:
```
# Try different values of K
k_values = range(1, 10)
inertias = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, n_init=10)
    kmeans.fit(Geo2)
    inertias.append(kmeans.inertia_)

# Plot the inertia values
plt.plot(k_values, inertias, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()
```



In [102]:
```
Kmeans = KMeans(n_clusters = 2,random_state=0, n_init=10).fit(Geo2)
Geo2
```

Out[10

		speed_limit	weather_conditio
407904	30	1	
407905	30	1	
407906	30	1	
407907	50	1	
407908	30	1	
...	...	...	
409607	30	1	
409608	30	1	
409609	30	1	
409610	70	1	
409611	30	1	

In [103]:
```
labels =Kmeans.predict(Geo2)
```
In [104]:
```
labels
```
Out[104]:
```
array([0, 0, 0, ..., 0, 1, 0])
```
In [106]:
```
centroids = Kmeans.cluster_centers_
centroids
```
Out[106]:
```
array([[30.52469136,  1.41589506],
       [60.13623978,  1.41689373]])
```
In [108]:
```
ig = plt.figure(figsize=(5, 5))

plt.scatter(Geo2['speed_limit'], Geo2['weather_conditions'], s=0.5, marker
plt.scatter(centroids[:, 0], centroids[:, 1], color='r', marker='*')

plt.xlabel('Speed Limit')
plt.ylabel('Weather Conditions')
plt.title('Scatter Plot of Speed Limit vs Weather Conditions')

plt.show()
```


In [ ]:

In [110]:
```
#increasing the number of clusters to 4.
Kmeans = KMeans(n_clusters = 4,random_state=0, n_init=10).fit(Geo2)
Geo2
```

Out[11

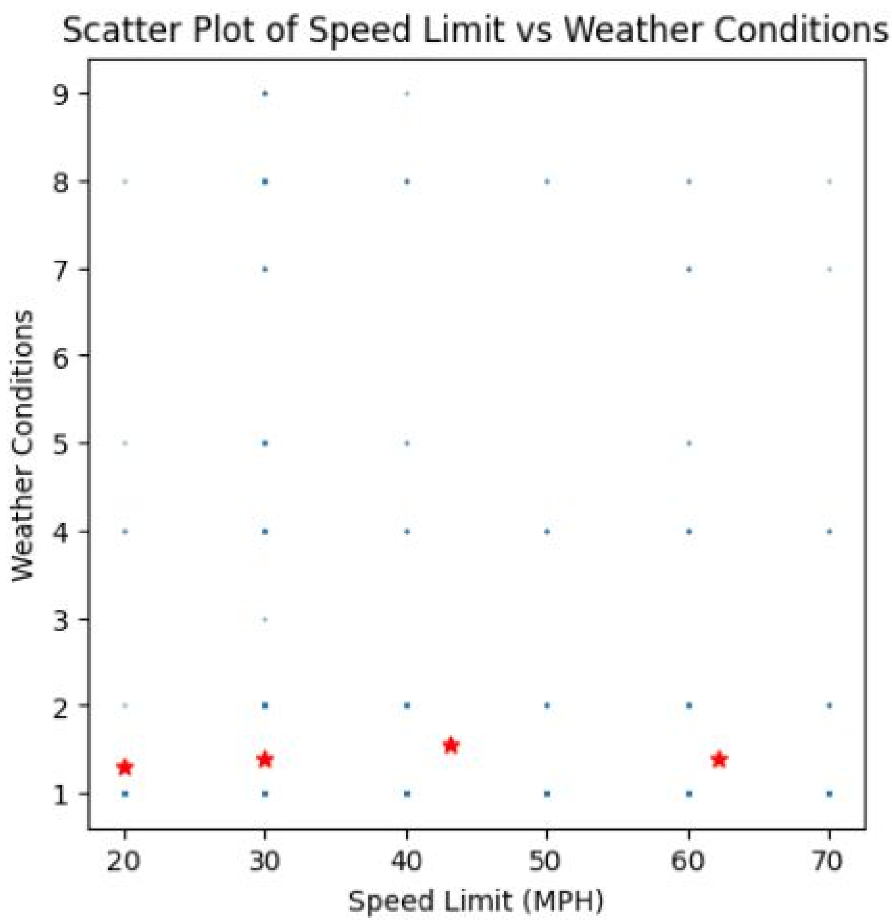
	speed_limit	weather_conditio
407904	30	1
407905	30	1
407906	30	1
407907	50	1
407908	30	1
...	...	...
409607	30	1
409608	30	1
409609	30	1
409610	70	1
409611	30	1

In [111]:
```
labels =Kmeans.predict(Geo2)
centroids = Kmeans.cluster_centers_
centroids
```

Out[111]:

```
array([[30.        ,  1.40441176],
       [62.19672131,  1.38688525],
       [43.1       ,  1.565     ],
       [20.        ,  1.3       ]])
```

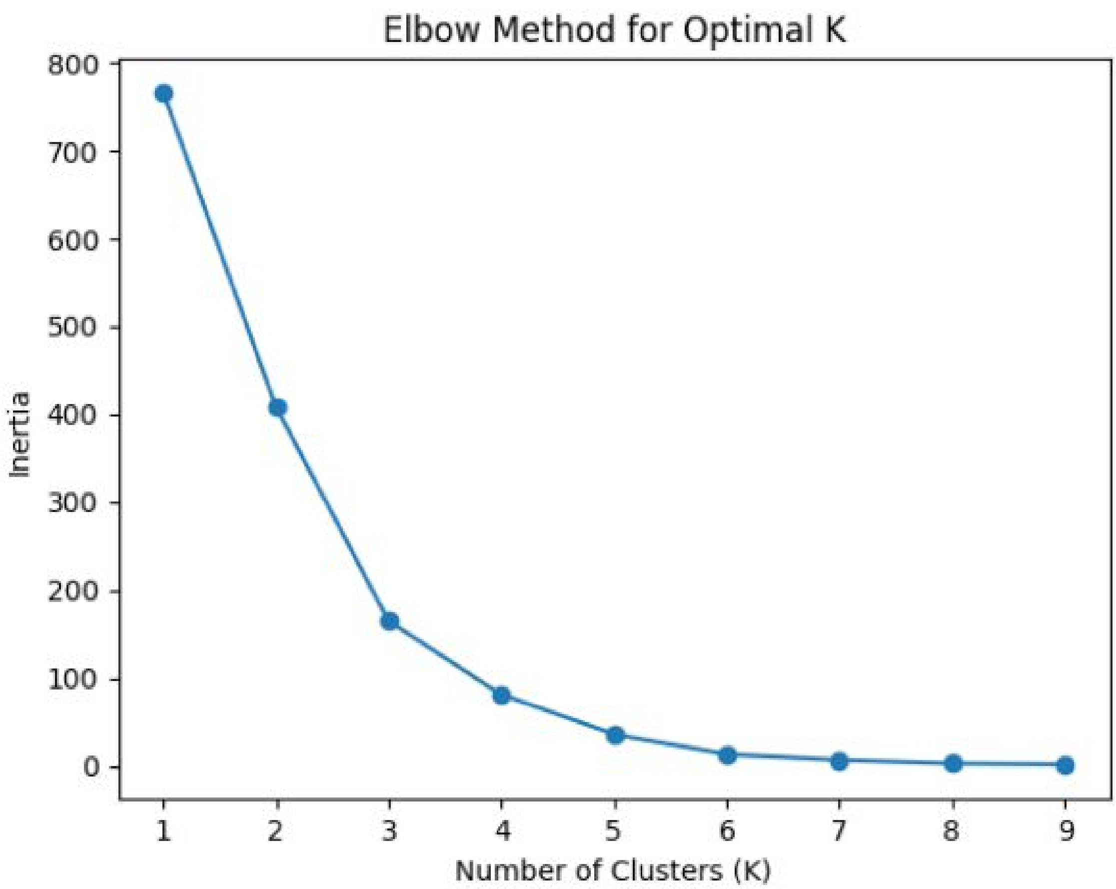
In [112]:
```
fig = plt.figure(figsize=(5,5))
plt.xlabel('Speed Limit (MPH)')
plt.ylabel('Weather Conditions')
plt.title('Scatter Plot of Speed Limit vs Weather Conditions')
plt.scatter(Geo2['speed_limit'], Geo2['weather_conditions'], s = 0.5, mark
plt.scatter(centroids[:,0], centroids[:,1], color='r', marker ="*")
plt.show()
```



Using k-means with 2 clusters with accident_severity and road_surface_conditions

In [128]:
```
Geo3 = humberside_ACC2020[["accident_severity", "road_surface_conditions"]]
```

In [129]:
```
# Try different values of K
k_values = range(1, 10)
inertias = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, n_init=10)
    kmeans.fit(Geo3)
    inertias.append(kmeans.inertia_)

# Plot the inertia values
plt.plot(k_values, inertias, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()
```

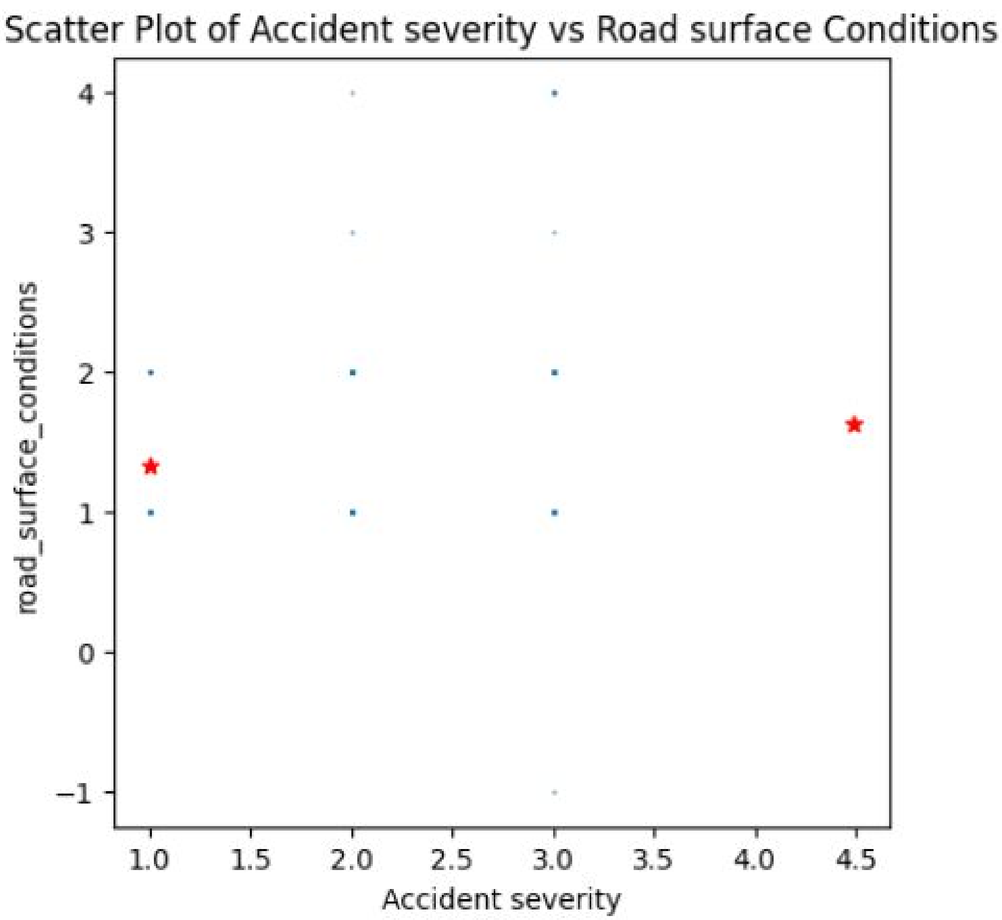


In [130]:
```
Kmeans = KMeans(n_clusters = 2,random_state=0, n_init=10).fit(Geo3)
Geo3
```

Out[13

	accident_severity	road_surface_conditio
407904	3	1
407905	3	1
407906	2	1
407907	3	1
407908	3	1
...	...	...
409607	3	1
409608	3	1
409609	3	1
409610	3	1
409611	3	1

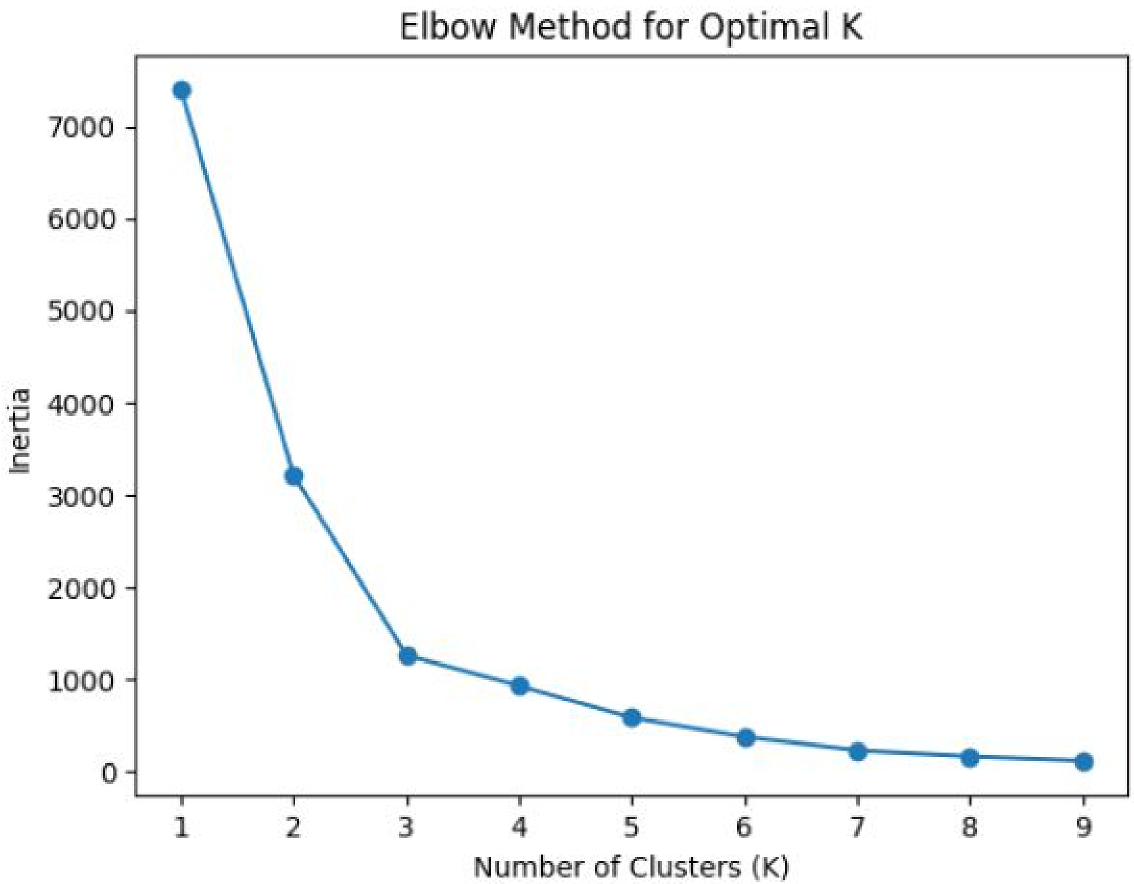
In [126]:
```
labels =Kmeans.predict(Geo3)
```
In [127]:
```
labels
```
Out[127]:
```
array([0, 0, 0, ..., 0, 1, 0])
```
In [131]:
```
centroids = Kmeans.cluster_centers_
centroids
```
Out[131]:
```
array([[2.71971496, 2.06175772],
       [2.78824477, 0.99838969]])
```
In [145]:
```
fig = plt.figure(figsize=(5,5))
plt.xlabel('Accident severity')
plt.ylabel('road_surface_conditions')
plt.title('Scatter Plot of Accident severity vs Road surface Conditions')
plt.scatter(Geo3['accident_severity'], Geo3['road_surface_conditions'], s :
plt.scatter(centroids[:,0], centroids[:,1], color='r', marker ="*")
plt.show()
```



Using k-means with 2 clusters with light_conditions and weather_conditions

In [171]:
```
Geo4 = humberside_ACC2020[["light_conditions", "weather_conditions"]]
```
In [172]:
```
# Try different values of K
k_values = range(1, 10)
inertias = []

for k in k_values:
```

```
        kmeans = KMeans(n_clusters=k, n_init=10)
        kmeans.fit(Geo4)
        inertias.append(kmeans.inertia_)

    # Plot the inertia values
    plt.plot(k_values, inertias, marker='o')
    plt.xlabel('Number of Clusters (K)')
    plt.ylabel('Inertia')
    plt.title('Elbow Method for Optimal K')
    plt.show()
```

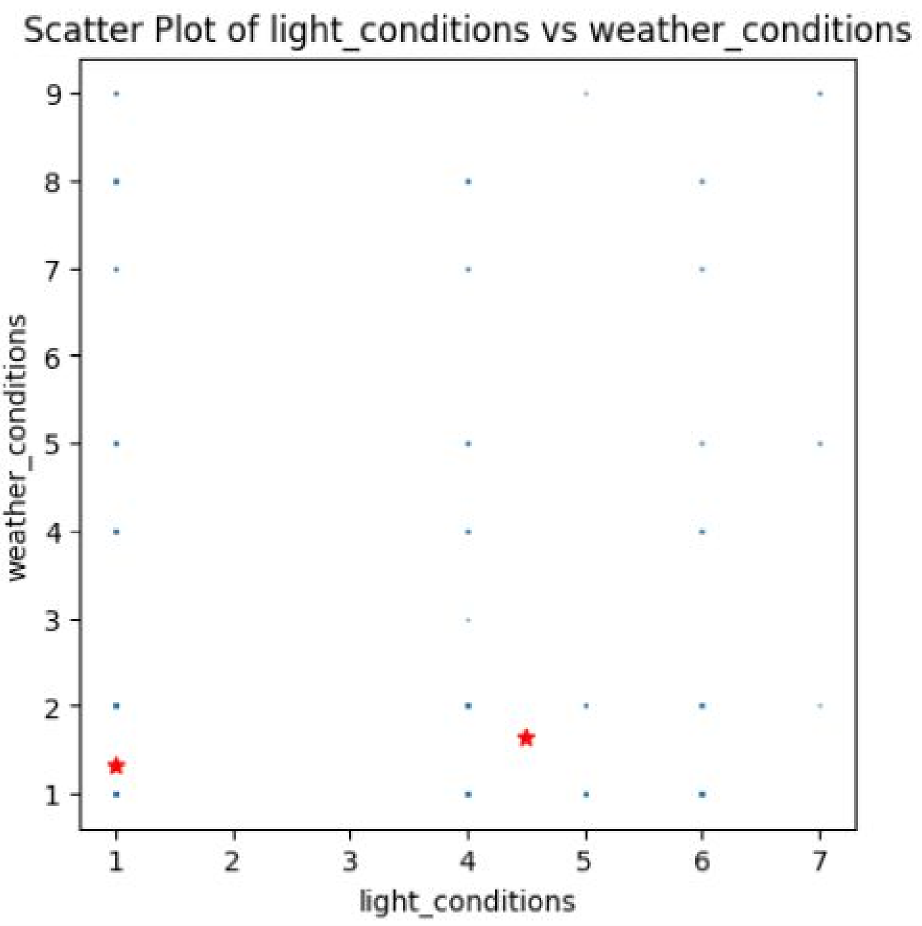


In [173]:
```
Kmeans = KMeans(n_clusters = 2,random_state=0, n_init=10).fit(Geo4)
Geo4
```

Out[17

		light_conditions	weather_conditio
407904	1	1	
407905	4	1	
407906	4	1	
407907	4	1	
407908	4	1	
...	...	...	
409607	1	1	
409608	4	1	
409609	1	1	
409610	1	1	
409611	4	1	

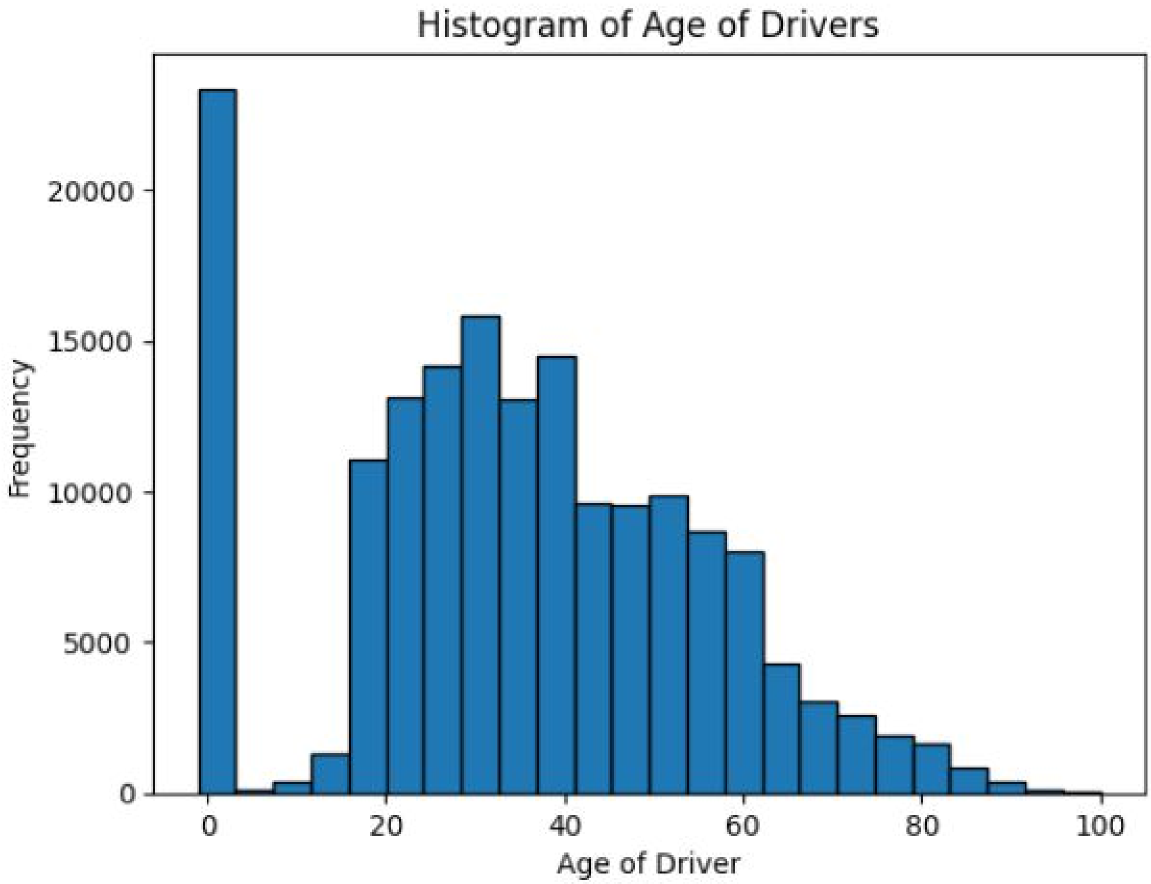
In [174]:
```
labels =Kmeans.predict(Geo4)
```
In [175]:
```
labels
```
Out[175]:
```
array([0, 1, 1, ..., 0, 0, 1])
```
In [176]:
```
centroids = Kmeans.cluster_centers_
centroids
```
Out[176]:
```
array([[1.        , 1.32656514],
       [4.48856549, 1.63617464]])
```
In [178]:
```
fig = plt.figure(figsize=(5,5))
plt.xlabel('light_conditions')
plt.ylabel('weather_conditions')
plt.title('Scatter Plot of light_conditions vs weather_conditions')
plt.scatter(Geo4['light_conditions'], Geo4['weather_conditions'], s = 0.5,
plt.scatter(centroids[:,0], centroids[:,1], color='r', marker ="*")
plt.show()
```



In [ ]:

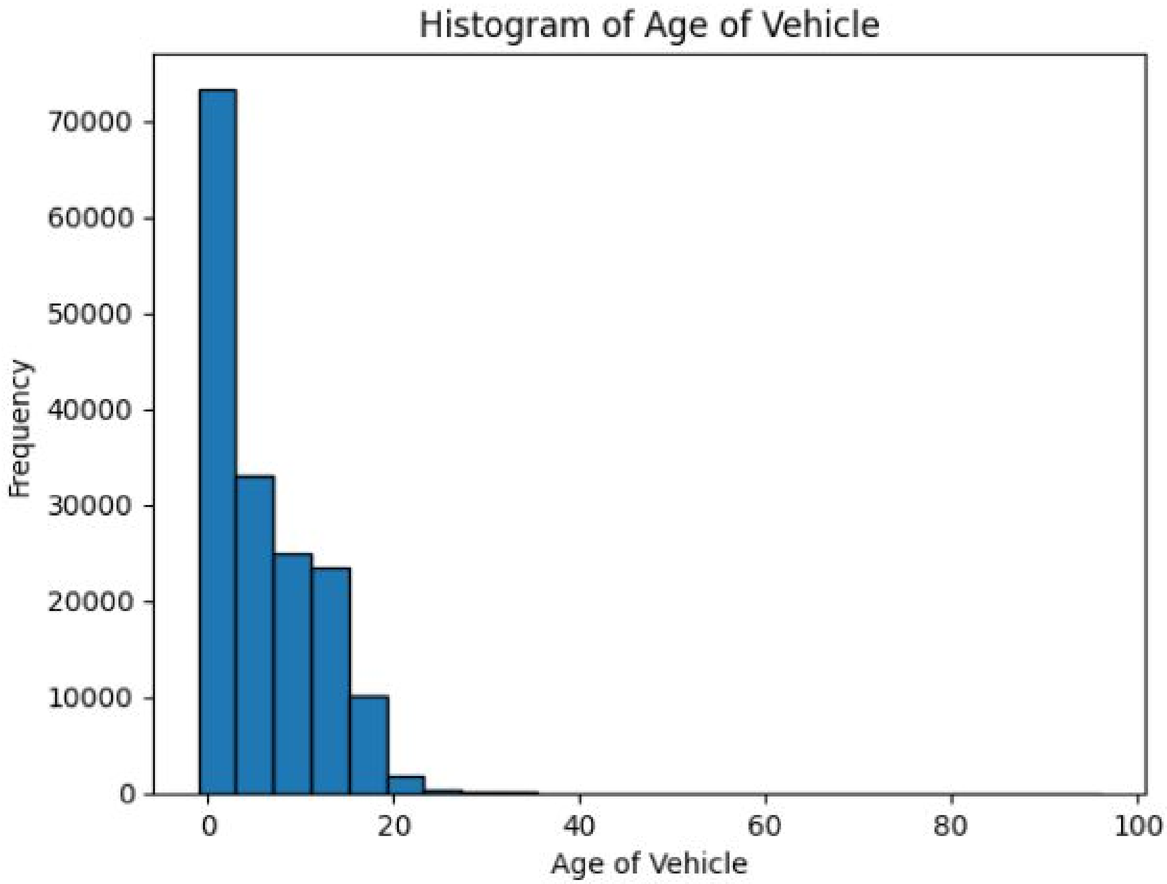
6. Using outlier detection methods, identify unusual entries in your data set. Should you keep these entries in your data?

In [181]:
```
import numpy as np
from scipy import stats
```
In [183]:
```
# Plot histograms of 'age_of_driver'
# Create the histogram of 'age_of_driver'
plt.hist(VEH2020['age_of_driver'], bins=24, edgecolor='black')

# Set the labels and title
plt.xlabel('Age of Driver')
plt.ylabel('Frequency')
plt.title('Histogram of Age of Drivers')

# Display the histogram
plt.show()
```



In [ ]:

In [184]:
```
# Create the histogram of 'age_of_vehicle'
plt.hist(VEH2020['age_of_vehicle'], bins=24, edgecolor='black')

# Set the labels and title
plt.xlabel('Age of Vehicle')
plt.ylabel('Frequency')
plt.title('Histogram of Age of Vehicle')

# Display the histogram
plt.show()
```



Vehicles where age_of_vehcle is -1

In [185]:
```
VEH2020[VEH2020['age_of_driver'] == -1]
```

Out[18

	vehi	accic	accic	accic	vehi	vehi	towi	vehi	vehi	vehi	...	jouri	sex_(	age_	age_	engi	prop	age_	gene	drive	d
6817	6817	2020	202C	0102	1	9	0	18	-1	-1	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
6817	6817	2020	202C	0102	2	9	0	2	0	0	...	6	3	-1	-1	1984	1	1	AUD Q5	-1	-
6817	6817	2020	202C	0102	1	9	0	18	7	3	...	6	1	-1	-1	-1	-1	-1	-1	-1	-
6817	6817	2020	202C	0102	2	9	9	99	9	9	...	6	3	-1	-1	122S	1	8	VAU COR	-1	-
6817	6817	2020	202C	0102	2	9	0	2	0	0	...	6	3	-1	-1	1984	1	9	VOLI GOLI	-1	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
848S	8489	2020	202C	9910	2	19	0	2	0	0	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
849C	8490	2020	202C	9910	2	19	0	2	0	0	...	6	3	-1	-1	2402	2	12	FORI TRAN	-1	-
849C	8490	2020	202C	9910	2	9	0	2	0	0	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
849C	8490	2020	202C	9910	3	9	0	2	0	0	...	6	3	-1	-1	-1	-1	-1	-1	-1	-
849C	8490	2020	202C	9910	4	9	0	2	0	0	...	6	3	-1	-1	-1	-1	-1	-1	-1	-

In [187]:
```
# Filter the records where age_of_driver is -1 in 'VEH2020'
filtered_vehicles = VEH2020[VEH2020['age_of_driver'] == -1]

# Extract the relevant columns from 'filtered_vehicles'
driver_info = filtered_vehicles[['age_of_driver', 'age_band_of_driver', 'a

# Merge 'driver_info' with 'casualty_df' using the common column 'accident_
result = CAS2020.merge(driver_info, on='accident_index')
# Select the desired columns
#print(result)

result
```

Out[18

	casu	accic	accic	accic	vehi	casu	casu	sex_	age_	age_	...	pede	pede	car_	bus_	pede	casu	casu	casu	age_	a
0	4847	2020	2020	0102	1	1	3	1	23	5	...	5	9	0	0	0	0	1	3	-1	-
1	4847	2020	2020	0102	1	1	1	1	62	9	...	0	0	0	0	0	9	1	6	-1	-
2	4847	2020	2020	0102	1	1	1	1	-1	-1	...	0	0	0	0	0	9	-1	-1	-1	-
3	4847	2020	2020	0102	1	1	1	1	30	6	...	0	0	0	0	0	9	1	2	-1	-
4	4847	2020	2020	0102	1	2	2	-1	-1	-1	...	0	0	1	0	0	9	1	1	-1	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2736	6002	2020	2020	9910	1	2	1	1	19	4	...	0	0	0	0	0	9	2	1	-1	-
2736	6002	2020	2020	9910	1	1	1	1	35	6	...	0	0	0	0	0	3	1	5	-1	-
2737	6003	2020	2020	9910	1	1	1	1	38	7	...	0	0	0	0	0	19	1	4	-1	-
2737	6003	2020	2020	9910	1	1	1	1	38	7	...	0	0	0	0	0	19	1	4	-1	-
2737	6003	2020	2020	9910	1	1	1	1	38	7	...	0	0	0	0	0	19	1	4	-1	-

Using Local Outlier Factor (LOF) for outliers detection in Humberside Region

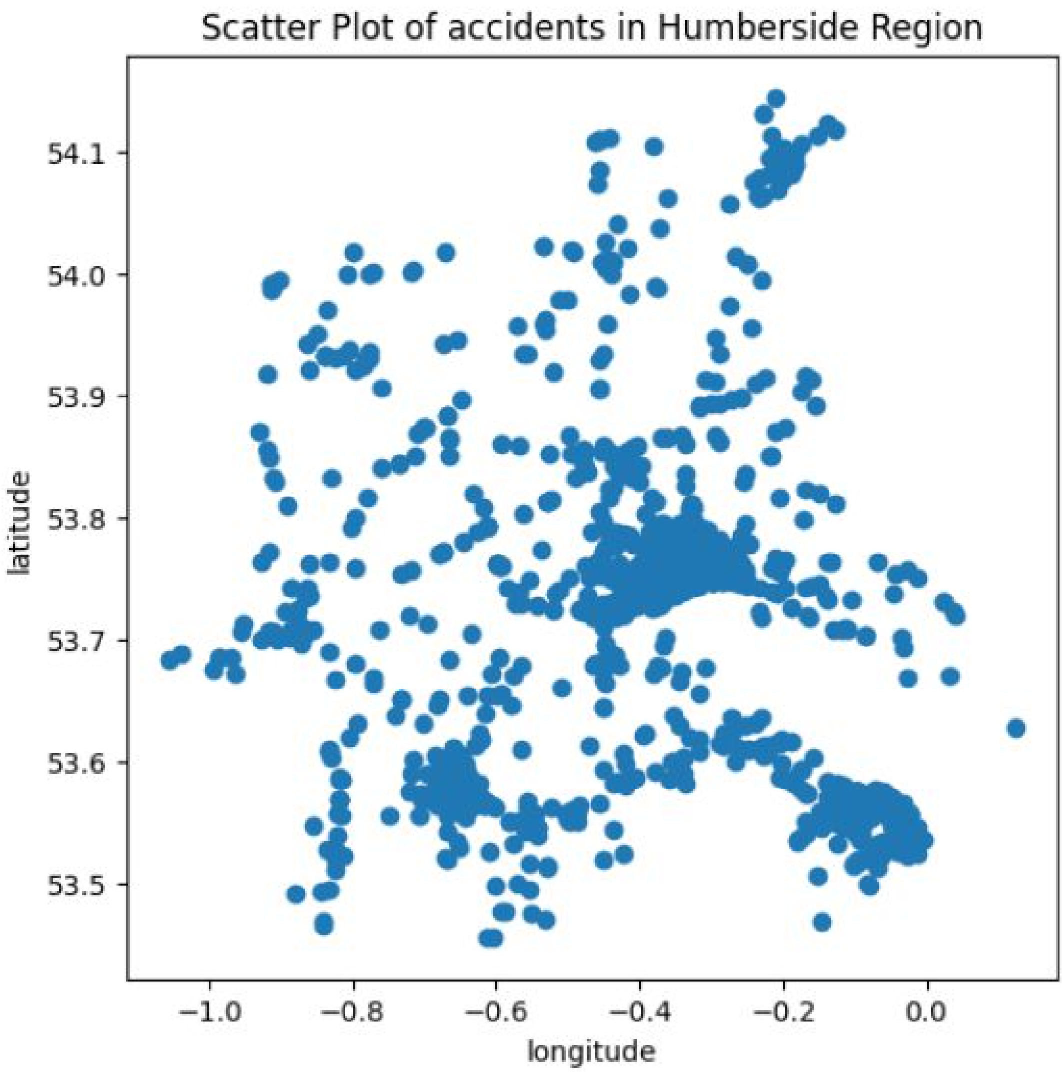
In [188]:
```
Geo = Geo.reset_index(drop=True)
Geo
```

Out[18

	longitude	latitu
0	-0.393424	53.744936
1	-0.528743	53.512895
2	-0.324858	53.791630
3	-0.095008	53.574501
4	-0.327733	53.767805
...	...	...
1658	-0.651104	53.566753
1659	-0.424674	53.839482
1660	-0.308880	53.782750
1661	-0.703181	53.569801
1662	-0.342063	53.742609

In [190]:
```
plt.figure(figsize=(6,6))
plt.scatter(Geo['longitude'],Geo['latitude'])

plt.xlabel('longitude')
plt.ylabel('latitude')
```

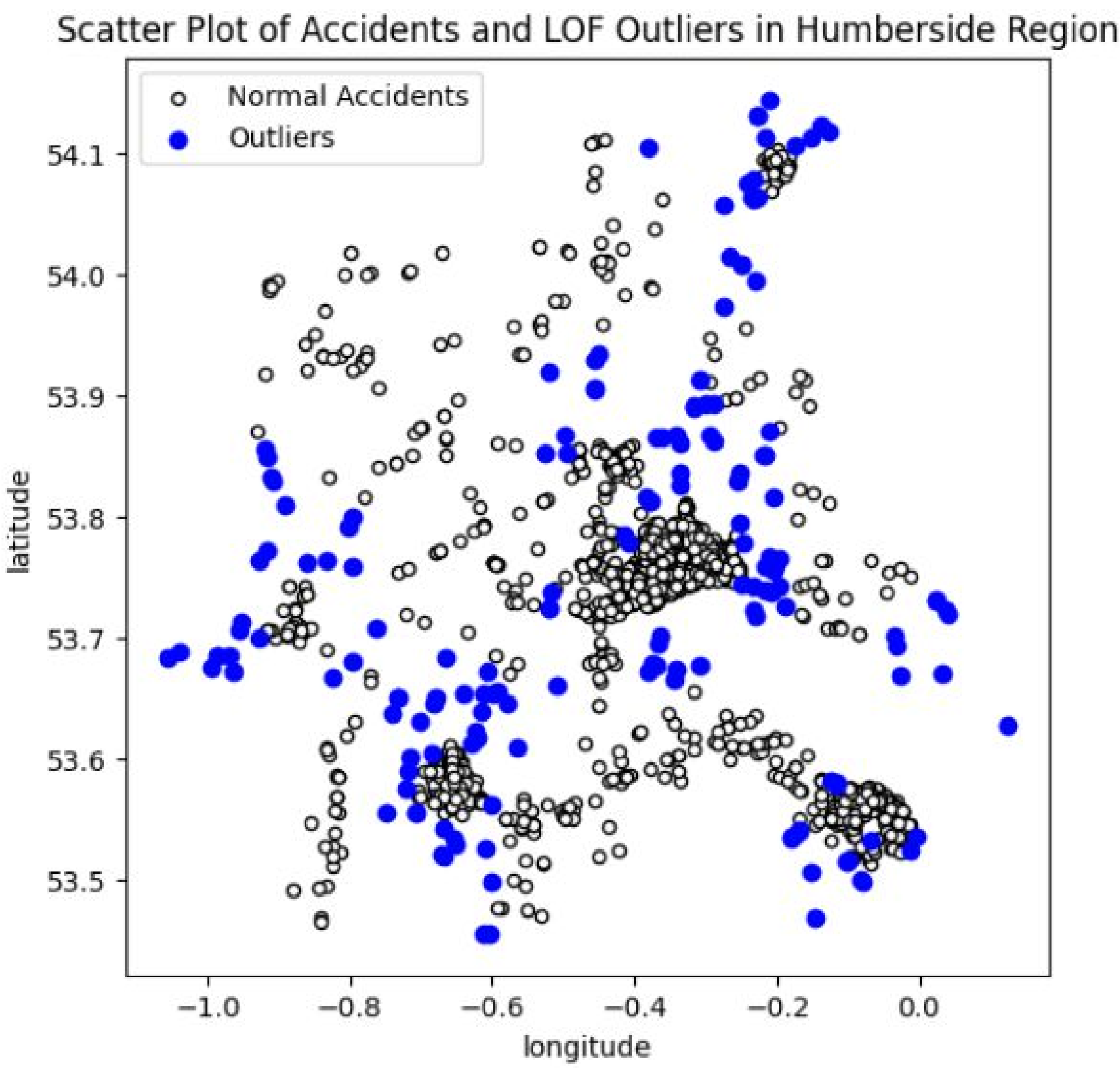
```
plt.title('Scatter Plot of accidents in Humberside Region')
plt.show()
```



In [191]:
```
from numpy import quantile,where
from sklearn.neighbors import LocalOutlierFactor
```

In [192]:
```
model = LocalOutlierFactor(n_neighbors=30, contamination=.1)
y_pred = model.fit_predict(Geo)
LOF_Scores = model.negative_outlier_factor_
LOF_pred=pd.Series(y_pred).replace([-1,1],[1,0])
LOF_anomalies=Geo[LOF_pred==1]
```

In [194]:
```
plt.figure(figsize=(6,6))
plt.scatter(Geo['longitude'], Geo['latitude'], c='white', s=20, edgecolor=
plt.scatter(LOF_anomalies['longitude'], LOF_anomalies['latitude'], c='blue
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.title('Scatter Plot of Accidents and LOF Outliers in Humberside Region
plt.legend()
plt.show()
```



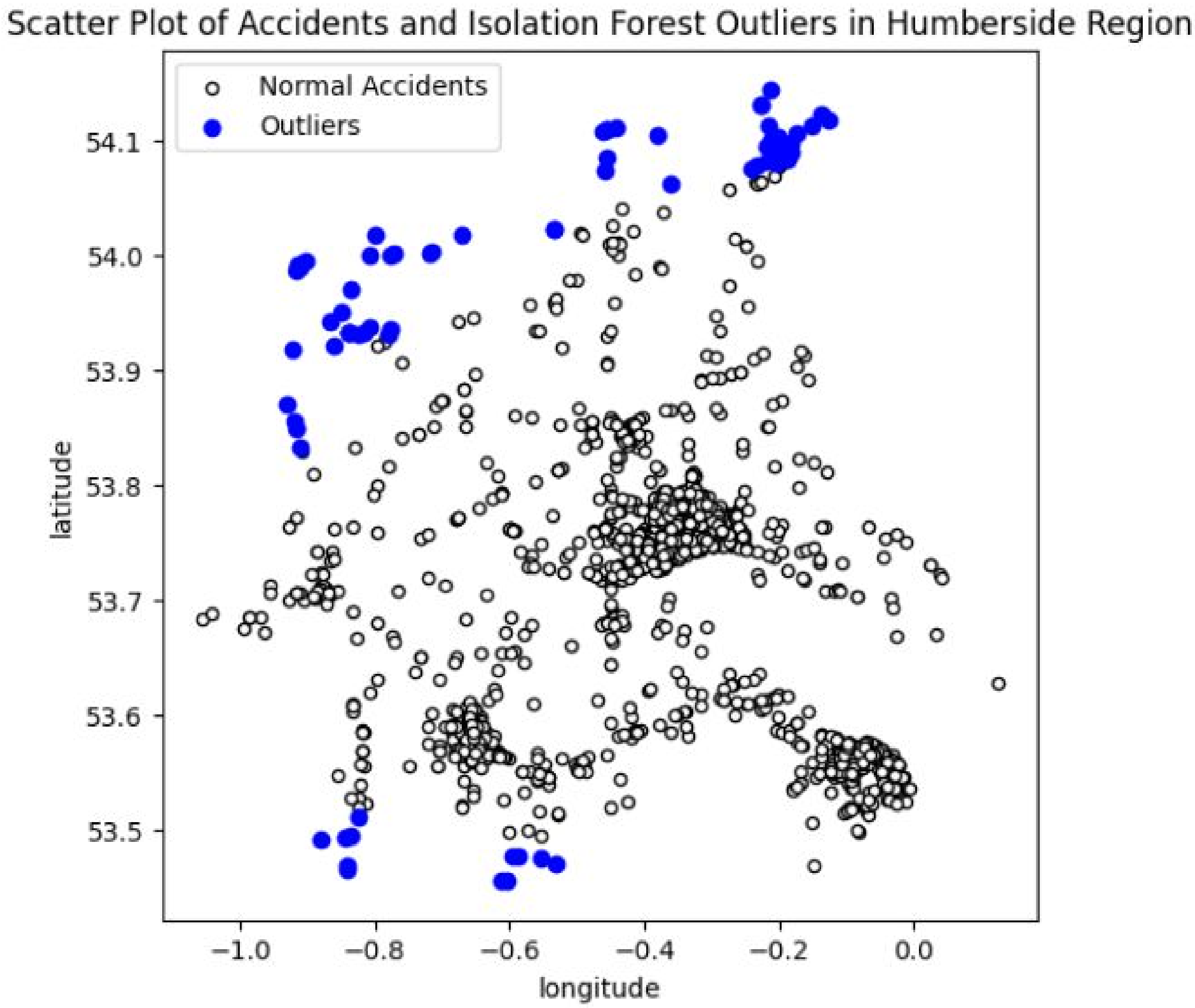
Isolation Forest for outliers detection in humberside region

In [195]:
```
from sklearn.ensemble import IsolationForest
```
In [200]:
```
ranst=np.random.RandomState(0)

model = IsolationForest(max_samples=100,random_state=ranst, contamination=
model.fit(Geo)
if_scores = model.decision_function(Geo)
if_anomalies=model.predict(Geo)
if_anomalies=pd.Series(if_anomalies).replace([-1,1],[1,0])
if_anomalies=Geo[if_anomalies==1];
```
In [201]:
```
if_anomalies
```

Out[20

	longitude	latitu
28	-0.837922	53.932848
45	-0.194271	54.091148
52	-0.612097	53.456973
72	-0.211803	54.143923
77	-0.182158	54.097510
...	...	...
1524	-0.186150	54.085938
1548	-0.594092	53.477707
1557	-0.219574	54.081251
1589	-0.825128	53.930143
1599	-0.532525	53.470207

In [203]:
```
plt.figure(figsize=(6,6))
plt.scatter(Geo['longitude'], Geo['latitude'], c='white', s=20, edgecolor=
plt.scatter(if_anomalies['longitude'], if_anomalies['latitude'], c='blue',
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.title('Scatter Plot of Accidents and Isolation Forest Outliers in Humb
plt.legend()
plt.show()
```



Local Outlier Factor (LOF) outlier detection for all accidents

In [204]:
```
Geo_all = ACC2020[["longitude", "latitude"]]
Geo_all = Geo_all.reset_index(drop=True)
Geo_all
```

Out[20

	longitude	latitu
0	-0.254001	51.462262
1	-0.139253	51.470327
2	-0.178719	51.529614
3	-0.001683	51.541210
4	-0.137592	51.515704
...	...	...
91194	-2.926320	56.473539
91195	-4.267565	55.802353
91196	-2.271903	57.186317
91197	-3.968753	55.950940
91198	-4.561040	56.003843

In [205]:
```
Geo_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 91199 entries, 0 to 91198
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   longitude  91185 non-null  float64
 1   latitude   91185 non-null  float64
dtypes: float64(2)
memory usage: 1.4 MB
```

In [221]:
```
Geo_all.replace(-1, np.nan)
```

Out[22

	longitude	latitu
370153	-0.254001	51.462262
370154	-0.139253	51.470327
370155	-0.178719	51.529614
370156	-0.001683	51.541210
370157	-0.137592	51.515704
...	...	...
461347	-2.926320	56.473539
461348	-4.267565	55.802353
461349	-2.271903	57.186317
461350	-3.968753	55.950940
461351	-4.561040	56.003843

In [ ]:

police_force

In [210]:

```
# Number of accidents in each region
ACC2020[ACC2020['longitude'].isna()]['police_force'].value_counts()
```

Out[210]:

```
63    8
5     1
7     1
12    1
13    1
52    1
62    1
Name: police_force, dtype: int64
```

Accident region using Police force

In [215]:

```
# Filter rows with missing longitude or latitude
missing_coords = ACC2020[ACC2020['longitude'].isnull() | ACC2020['latitude

# Calculate median longitude, latitude, and mode of rural_or_urban for eac
police_forces_to_calculate = missing_coords['police_force'].unique()
police_force_data = ACC2020[ACC2020['police_force'].isin(police_forces_to_
median_coordinates = police_force_data.groupby('police_force')[['longitude
mode_rural_or_urban = police_force_data.groupby('police_force')['urban_or_

# Update missing longitude, latitude, and rural_or_urban based on police_f
for index, row in missing_coords.iterrows():
    police_force = row['police_force']
    if police_force in median_coordinates.index:
        ACC2020.at[index, 'longitude'] = median_coordinates.loc[police_for
        ACC2020.at[index, 'latitude'] = median_coordinates.loc[police_forc
```

```
            ACC2020.at[index, 'location_easting_osgr'] = median_coordinates.lo
            ACC2020.at[index, 'location_northing_osgr'] = median_coordinates.l
            ACC2020.at[index, 'urban_or_rural_area'] = mode_rural_or_urban[pol
```

In [216]:

```
ACC2020['urban_or_rural_area'].value_counts()
```

Out[216]:

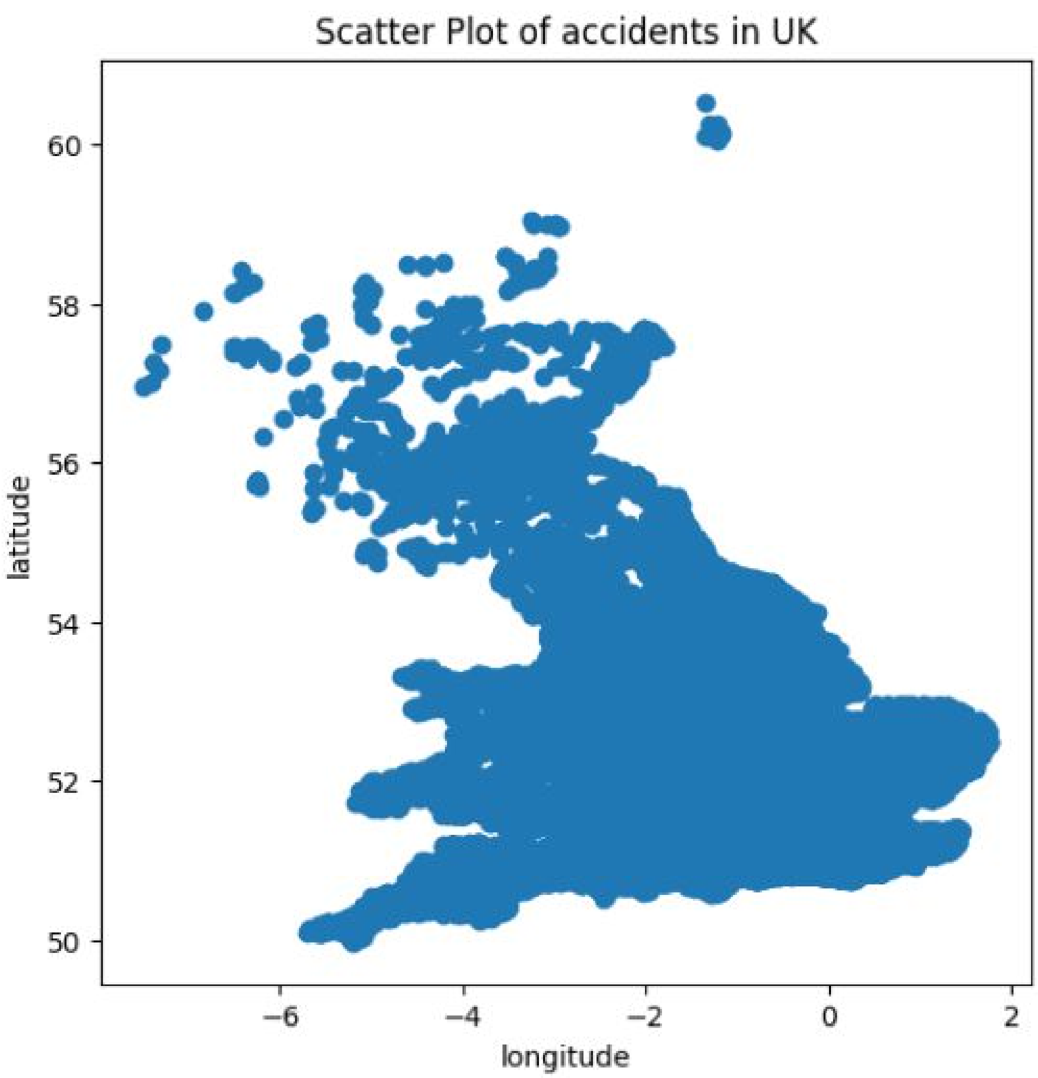
```
1    61742
2    29457
Name: urban_or_rural_area, dtype: int64
```

In [217]:

```
# Create Data frame from longitude and latitude of accident_df
Geo_all = ACC2020[['longitude', 'latitude']]
```

In [218]:

```
plt.figure(figsize=(6,6))
plt.scatter(Geo_all['longitude'],Geo_all['latitude'])
plt.xlabel('longitude')
plt.ylabel('latitude')
plt.title('Scatter Plot of accidents in UK')
plt.show()
```



In [ ]:

7. Can you develop a classification model using the provided data that accurately predicts fatal injuries sustained in road traffic accidents, with the aim of informing and improving road safety measures?

In [220]:
```
DATA_NEW.head()
```

Out[22

	accic	num	num	road	spee	light	weat	road	polic	vehic	sex_c	age_	engi	age_	casu	sex_c	pede	Regi	lof_o	km
8349	3	2	1	6	30	1	1	1	16	9	1	24	1248	11	1	1	0	Hull	1	1
8349	3	2	1	6	30	1	1	1	16	19	1	48	1968	5	1	1	0	Hull	1	1
8349	3	2	1	3	30	1	1	2	16	9	2	34	1997	4	1	1	0	Hull	1	1
8349	3	2	1	3	30	1	1	2	16	1	1	61	-1	-1	1	1	0	Hull	1	1

In [221]:
```
DATA_NEW.isnull().sum()
```

Out[221]:

```
accident_severity          0
number_of_vehicles         0
number_of_casualties       0
road_type                  0
speed_limit                0
light_conditions           0
weather_conditions         0
road_surface_conditions    0
police_force               0
vehicle_type               0
sex_of_driver              0
age_of_driver              0
engine_capacity_cc         0
age_of_vehicle             0
casualty_class             0
sex_of_casualty            0
pedestrian_location        0
Region                     0
lof_outlier                0
kmeans_labels              0
dtype: int64
```

In [ ]:

In [222]:
```
DATA_NEW.accident_severity.value_counts()
```

Out[222]:

```
3    3119
2     792
1      74
Name: accident_severity, dtype: int64
```

In [224]:
```
X = DATA_NEW.drop(["accident_severity", 'Region', 'lof_outlier', 'kmeans_l
y = DATA_NEW.accident_severity
```

In [ ]:

In [225]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
```

In [ ]:

In [226]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClass
```

Gradient Boosting Classifier

In [229]:
```
gb = GradientBoostingClassifier()

gb.fit(x_train, y_train)
```

Out[22

In [231]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [232]:
```
gb_pred = gb.predict(x_test)
```

In [233]:
```
gb_pred
```

Out[233]:

```
array([3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 2, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 2,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 2,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3,
       3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3,
       3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 1, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2,
       3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 2, 3, 3, 3, 3, 3, 3,
       2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3,
       1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3,
       3, 2, 3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 2, 3,
       3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3,
       3, 2, 3, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2,
       2, 3, 3, 3, 3], dtype=int64)
```

In [321]:
```
confusion = confusion_matrix(y_test, gb_pred)
```

In [322]:
```
confusion
```

Out[322]:

```
array([[  4,   1,  10],
       [  0,  30, 128],
       [  3,  16, 605]], dtype=int64)
```

In [ ]:

In [244]:
```
C_Report = classification_report(y_test, gb_pred, target_names=["Fatal", "
```

In [245]:
```
C_Report
```

Out[245]:

```
'              precision    recall  f1-score   support\n\n       Fatal
0.57      0.27      0.36        15\n     Seriuos       0.64      0.19      0.2
9       158\n      Slight       0.81      0.97      0.89       624\n\n    accur
acy                           0.80       797\n   macro avg       0.67      0.4
8      0.51       797\nweighted avg       0.77      0.80      0.76       797\n'
```

In [ ]:

In [ ]: