Data Mgt 6305 Group 10 Final Project - UK Car Accident Data from 2014 to 2016

Ask 1

Identify and Describe the Dataset

The group has decided to use Road Traffic Collision dataset for the project. This data contains two distinct files with information about road accidents in the United Kingdom from 2005 and 2016. The first file (accident_data.csv file) consists of columns regarding the accident identification number, the class of road on which the accident occurred, the date of the accident, and weather conditions, among other variables, which can be accessed through the link below. The second file (vehicle_data.csv file) contains information about the type of vehicle involved in the accident. It is a CSV file with about 1.4 million records and a size of 3003.3 MB for the accident_data.csv file and 1.6 million records, 644.4 MB for the vehicle_data.csv file. We obtained the data from Kaggle.com through this link: https://www.kaggle.com/datasets/salmankhaliq22/road-traffic-collision-dataset?select=vehicle_data.csv

Why this data is important and what's appealing about it

Literature about death tolls through road carnage in the UK points to a low record since 1960 (refer to the first paragraph of 'Road Traffic Collision Dataset' via the link above). However, recent events reveal a sharp contrast to this literature. It is interesting to note that 1,516 people lost their lives in the UK in 2020 through road accidents. This observation caught the eye of the group members hence our decision to analyze the data and come up with the causal effects of the recent increase in road accidents and their attendant death tolls in the United Kingdom.

Given time constrain regarding the project, the group decided to use this dataset because it has reasonable attributes and description to perform our dimensional modeling and analysis; notwithstanding the null values, many columns have numeric values, which we were able to clean up to make the data tidy for our analysis. Additionally, we believe our project will add to the existing literature on accidents in the UK and the suggested solutions we have offered to guide stakeholders who may come across it for future reference.

Is it suitable for dimensional modeling and analytical analysis?

We used csvkit to access the columns associated with the various files. This helped us to identify the null values and fill them. It also helped us to identify the columns that will help us to formulate our analytical questions. Based on this, we dropped the columns we deemed

redundant. For example, in the Accident table we have dropped columns which has mostly None/Null values like 1st Road Class, Second Road Class etc. In the Vehicle table we dropped columns specific to the engine information of the vehicle and if the vehicle was left-hand drive or not due to irrelevancy and high number of nulls. We also notice that the years on which the data for the two files were captured vary. So, we decided to do our analysis based on the years from 2014-2016.

Below is the Data Dictionary for Accident and Vehicle Data.

Accident Data	Column Description
Accident_Severity	the target variable. Indicates 3 classes of severity: "slight," "serious" and "fatal."
Did_Police_Officer_Attend_Scene_of_Accident	3 options: 1 - Yes. 2 - No. 3 - No, the accident was reported by a self-complaint
LSOA_of_Accident_Location	Lower Layer Super Output Area of accident
Number_of_Casualties	number of those killed or injured in the accident
Number_of_Vehicles	number of vehicles involved in the accident
1st_Road_Class	road class of 1st road the accident happened on. For more information on UK road classes
2nd_Road_Class	road class of 2nd road the accident happened on.
Carriageway_Hazards	an observation of any hazards in the road at the time of the accident eg. animals or pedestrians in the road.
Junction_Control	what controls are in place to control traffic at the a junction.
Junction_Detail	what type of junction at the location of the accident
Light_Conditions	the light condition at the time of the accident.
Pedestrian_Crossing-Human_Control	was there a human controlled crossing present at the scene of the accident.
Pedestrian_Crossing-Physical_Facilities	number of vehicles involved in the accident
Road_Surface_Conditions	condition of the road when the accident took place.
Road_Type	Type of Road
Special_Conditions_at_Site	was there any other factors which could have caused the accident ie. oil on the road, faulty traffic lights etc.

Speed_limitsort	speed limit where the accident took place.
Urban_or_Rural_Area	was the location rural or urban.
Weather_Conditions	what was the weather like at the time of the accident.
1st_Road_Number	the road number of the 1st road the accident happened on
2nd_Road_Number	road number of 2nd road the accident happened on
navigation Latitude	latitude of where the accident took place
Local Authority_(District)	which district council jurisdiction did the accident occur.
Local Authority_(Highway)	who is the highway authority for the area the accident took place.
Location_Easting_OSGR	easting grid reference
Location_Northing_OSGR	northing grid reference.
navigationLongitude	longitude of where the accident took place
Police_Force	the police force responsible for the area.
checkInScotland	did the accident take place in Scotland
Accident_Index	accident ID
Date	date of the accident.
Day_of_Week	Day the accident happened
calendar_todayTime	the time the accident took place.
Year	the year the accident took place.

Vehicle Data	Column Description
Accident_Index	accident ID
Age_Band_of_Driver	Range of Age of the Drivers
Age_of_Vehicle	How old the Vehicle was
Driver_Home_Area_Type	Type of the home driver is living in
Driver_IMD_Decile	Driver's Index of Multiple Deprivation
Engine_Capacity_.CC.	Capacity of Car's Engine in CC
Hit_Object_in_Carriageway	Did the person hit by an object in carriageway
Hit_Object_off_Carriageway	Did the person hit by an object off carriageway
Journey_Purpose_of_Driver	Purpose of the Journey of the driver
Junction_Location	Location of the Junction
make	Make of the car
model	Model of the car
Propulsion_Code	propulsion code
Sex_of_Driver	Gender of the driver
Skidding_and_Overturning	whether car was skidding or overturning
Towing_and_Articulation	Information about towing and articulation
Vehicle_Leaving_Carriageway	Did vehicle leave carriageway
Vehicle_Location.Restricted_Lane	was vehicle in restricted lane?
Vehicle_Manoeuvre	Maneuver of vehicle
Vehicle_Reference	Reference number of vehicle in accident
Vehicle_Type	Type of vehicle
Was_Vehicle_Left_Hand_Drive	Was vehicle left hand drive?
1st_Point_of_Impact	POI

Analytical Questions

(1) Cities with maximum number of accidents according to severity

(2) Most accident prone car type

(3) Day/time and age bracket of drivers that saw most accidents

Describe any concerns with the data and changes you expect to overcome

The data consists of two distinct files with a lot of null values. This poses challenges in dealing with the null values, especially columns with categorical attributes (LSOA_of_Accident_Location, 2nd_Road_Class, Junction Control, among other variables) have been dropped, and a new dictionary has been created to accommodate the columns we used in the creation of the Tables.

Due to the large size of the dataset, we were not able to perform analysis to capture every aspect of the data. Therefore, we filtered relevant columns to enable us to satisfy our analytical questions, which primarily is to find out the highest accidents and the causal effects.

Ask 2: Data Wrangling and Dimensional Modeling

Accident Data CSV Wrangling

In [1]:
```
import os
os.environ['SPARK_HOME'] = '/usr/local/lib/spark'
import findspark
findspark.init()
from pyspark import SparkContext
spark = SparkContext(appName='project-final')
spark
from pyspark import SQLContext
sqlc = SQLContext(spark)
sqlc
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(new
Level).
22/12/09 18:21:53 WARN NativeCodeLoader: Unable to load native-hadoop library for you
r platform... using builtin-java classes where applicable
/usr/local/lib/spark/python/pyspark/sql/context.py:112: FutureWarning: Deprecated in
3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
```

Out[1]:
```
<pyspark.sql.context.SQLContext at 0x7f632c1bc490>
```

In [2]:
```
%load_ext sql
```

Uploading the data using wget file

In [3]:
```
!wget https://pbfinalproject.s3.amazonaws.com/Accident_2014_2016.csv
```

```
--2022-12-09 18:21:55--  https://pbfinalproject.s3.amazonaws.com/Accident_2014_2016.c
sv
Resolving pbfinalproject.s3.amazonaws.com (pbfinalproject.s3.amazonaws.com)... 54.23
1.204.33, 52.217.230.129, 52.217.0.35, ...
Connecting to pbfinalproject.s3.amazonaws.com (pbfinalproject.s3.amazonaws.com)|54.23
1.204.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 132966597 (127M) [text/csv]
Saving to: ‘Accident_2014_2016.csv.6’

Accident_2014_2016. 100%[===================>] 126.81M  60.3MB/s    in 2.1s

2022-12-09 18:21:57 (60.3 MB/s) - ‘Accident_2014_2016.csv.6’ saved [132966597/1329665
97]
```

In [4]:
```
!wc -l Accident_2014_2016.csv
```

```
423000 Accident_2014_2016.csv
```

In [5]:
```
!csvcut -n Accident_2014_2016.csv
```

```
  1:
  2: Accident_Index
  3: 1st_Road_Class
  4: 1st_Road_Number
  5: 2nd_Road_Class
  6: 2nd_Road_Number
  7: Accident_Severity
  8: Carriageway_Hazards
  9: Date
 10: Day_of_Week
 11: Did_Police_Officer_Attend_Scene_of_Accident
 12: Junction_Control
 13: Junction_Detail
 14: Latitude
 15: Light_Conditions
 16: Local_Authority_(District)
 17: Local_Authority_(Highway)
 18: Location_Easting_OSGR
 19: Location_Northing_OSGR
 20: Longitude
 21: LSOA_of_Accident_Location
 22: Number_of_Casualties
 23: Number_of_Vehicles
 24: Pedestrian_Crossing-Human_Control
 25: Pedestrian_Crossing-Physical_Facilities
 26: Police_Force
 27: Road_Surface_Conditions
 28: Road_Type
 29: Special_Conditions_at_Site
 30: Speed_limit
 31: Time
 32: Urban_or_Rural_Area
 33: Weather_Conditions
 34: Year
 35: InScotland
```

Keeping only the below columns for our analysis using csvcut

In [6]:
```
!csvcut -c 2,7,8,9,10,11,14,15,16,20,22,23,26,27,30,31,32,33,34 Accident_2014_2016.csv
```

In [7]:
```
!csvcut -n Accident_2014_2016_updated.csv
```

```
  1: Accident_Index
  2: Accident_Severity
  3: Carriageway_Hazards
  4: Date
  5: Day_of_Week
  6: Did_Police_Officer_Attend_Scene_of_Accident
  7: Latitude
  8: Light_Conditions
  9: Local_Authority_(District)
 10: Longitude
 11: Number_of_Casualties
 12: Number_of_Vehicles
 13: Police_Force
 14: Road_Surface_Conditions
 15: Speed_limit
 16: Time
 17: Urban_or_Rural_Area
 18: Weather_Conditions
 19: Year
```

In [8]:
```
!head -n 100000 Accident_2014_2016_updated.csv | csvstat
```

```
/home/ubuntu/.local/lib/python3.8/site-packages/agate/table/from_csv.py:74: RuntimeWa
rning: Error sniffing CSV dialect: Could not determine delimiter
  1. "Accident_Index"

        Type of data:          Text
        Contains null values:  False
        Unique values:         99999
        Longest value:         13 characters
        Most common values:    201401BS70001 (1x)
                               201401BS70002 (1x)
                               201401BS70003 (1x)
                               201401BS70004 (1x)
                               201401BS70006 (1x)

  2. "Accident_Severity"

        Type of data:          Text
        Contains null values:  False
        Unique values:         3
        Longest value:         7 characters
        Most common values:    Slight (85920x)
                               Serious (13049x)
                               Fatal (1030x)

  3. "Carriageway_Hazards"

        Type of data:          Text
        Contains null values:  True (excluded from calculations)
        Unique values:         6
        Longest value:         47 characters
        Most common values:    None (98479x)
                               Other object on road (520x)
                               Any animal in carriageway (except ridden horse) (420x)
                               Pedestrian in carriageway - not injured (368x)
                               Previous accident (111x)

  4. "Date"

        Type of data:          Date
        Contains null values:  False
        Unique values:         365
        Smallest value:        2014-01-01
        Largest value:         2014-12-31
        Most common values:    2014-12-05 (408x)
                               2014-10-10 (387x)
                               2014-11-14 (384x)
                               2014-12-16 (373x)
                               2014-11-04 (372x)

  5. "Day_of_Week"

        Type of data:          Date
        Contains null values:  False
        Unique values:         7
        Smallest value:        0001-01-02
        Largest value:         0001-01-08
        Most common values:    0001-01-05 (16442x)
```

```
                          0001-01-03 (15282x)
                          0001-01-02 (15202x)
                          0001-01-04 (14863x)
                          0001-01-08 (14482x)

6. "Did_Police_Officer_Attend_Scene_of_Accident"

      Type of data:          Number
      Contains null values:  False
      Unique values:         2
      Smallest value:        1
      Largest value:         2
      Sum:                   118497
      Mean:                  1.185
      Median:                1
      StDev:                 0.388
      Most common values:    1 (81501x)
                             2 (18498x)

7. "Latitude"

      Type of data:          Number
      Contains null values:  False
      Unique values:         93094
      Smallest value:        51.296
      Largest value:         55.791
      Sum:                   5264818.552
      Mean:                  52.649
      Median:                52.542
      StDev:                 1.025
      Most common values:    51.514 (10x)
                             51.513 (9x)
                             51.513 (8x)
                             51.504 (7x)
                             51.516 (7x)

8. "Light_Conditions"

      Type of data:          Text
      Contains null values:  False
      Unique values:         5
      Longest value:         27 characters
      Most common values:    Daylight (73288x)
                             Darkness - lights lit (21062x)
                             Darkness - no lighting (3999x)
                             Darkness - lighting unknown (1098x)
                             Darkness - lights unlit (552x)

9. "Local_Authority_(District)"

      Type of data:          Text
      Contains null values:  False
      Unique values:         238
      Longest value:         28 characters
      Most common values:    Birmingham (2628x)
                             Leeds (1935x)
                             Westminster (1597x)
```

```
                            Lambeth (1263x)
                            Bradford (1246x)
```

10. "Longitude"

```
       Type of data:          Number
       Contains null values:  False
       Unique values:         93989
       Smallest value:        -3.6
       Largest value:         1.759
       Sum:                   -101009.454
       Mean:                  -1.01
       Median:                -1.092
       StDev:                 1.068
       Most common values:    -0.104 (11x)
                              -0.089 (9x)
                              -0.042 (8x)
                              -0.192 (7x)
                              -0.136 (7x)
```

11. "Number_of_Casualties"

```
       Type of data:          Number
       Contains null values:  False
       Unique values:         22
       Smallest value:        1
       Largest value:         93
       Sum:                   132570
       Mean:                  1.326
       Median:                1
       StDev:                 0.872
       Most common values:    1 (78518x)
                              2 (14741x)
                              3 (4244x)
                              4 (1545x)
                              5 (590x)
```

12. "Number_of_Vehicles"

```
       Type of data:          Number
       Contains null values:  False
       Unique values:         12
       Smallest value:        1
       Largest value:         13
       Sum:                   183465
       Mean:                  1.835
       Median:                2
       StDev:                 0.68
       Most common values:    2 (61918x)
                              1 (28701x)
                              3 (7343x)
                              4 (1551x)
                              5 (338x)
```

13. "Police_Force"

```
       Type of data:          Text
```

```
       Contains null values:  False
       Unique values:         30
       Longest value:         19 characters
       Most common values:    Metropolitan Police (25682x)
                              West Midlands (5603x)
                              West Yorkshire (5043x)
                              Essex (4117x)
                              Greater Manchester (4004x)
```

14. "Road_Surface_Conditions"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         6
       Longest value:         28 characters
       Most common values:    Dry (71355x)
                              Wet or damp (27118x)
                              Frost or ice (1072x)
                              Data missing or out of range (160x)
                              Flood over 3cm. deep (158x)
```

15. "Speed_limit"

```
       Type of data:          Number
       Contains null values:  False
       Unique values:         6
       Smallest value:        20
       Largest value:         70
       Sum:                   3701600
       Mean:                  37.016
       Median:                30
       StDev:                 13.062
       Most common values:    30 (69379x)
                              60 (11323x)
                              40 (7652x)
                              70 (5845x)
                              50 (3654x)
```

16. "Time"

```
       Type of data:          TimeDelta
       Contains null values:  False
       Unique values:         1437
       Smallest value:        0:00:01
       Largest value:         0:23:59
       Sum:                   973 days, 22:02:02
       Most common values:    0:17:00 (958x)
                              0:18:00 (889x)
                              0:17:30 (883x)
                              0:15:30 (868x)
                              0:16:00 (827x)
```

17. "Urban_or_Rural_Area"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         2
```

```
        Longest value:          5 characters
        Most common values:     Urban (71385x)
                                Rural (28614x)
```

18. "Weather_Conditions"

```
        Type of data:           Text
        Contains null values:   False
        Unique values:          9
        Longest value:          21 characters
        Most common values:     Fine no high winds (82196x)
                                Raining no high winds (11335x)
                                Unknown (1777x)
                                Other (1679x)
                                Raining + high winds (1308x)
```

19. "Year"

```
        Type of data:           Number
        Contains null values:   False
        Unique values:          1
        Smallest value:         2014
        Largest value:          2014
        Sum:                    201397986
        Mean:                   2014
        Median:                 2014
        StDev:                  0
        Most common values:     2014 (99999x)
```

Row count: 99999

Now checking columns for some missing values or nulls using Relational Databases

In [9]:
```
accident = sqlc.read.csv("Accident_2014_2016_updated.csv",header='true')
```

We are changing the name of the column as below because it will create trouble while cleaning the data

In [10]:
```
accident = accident.withColumnRenamed("Local_Authority_(District)", "Local_Authority_D
```

In [11]:
```
accident.count()
```

Out[11]: 422999

Checking each column for Data missing or null

In [12]:
```
accident.groupBy("Accident_Index").count().orderBy("count", ascending=False).show(10)
```

```
[Stage 4:============================>                            (1 + 1) / 2]
```

```
+--------------+-----+
|Accident_Index|count|
+--------------+-----+
| 201401BS70305|    1|
| 201401BS70336|    1|
| 201401BS70416|    1|
| 201401BS70435|    1|
| 201401BS70451|    1|
| 201401BS70790|    1|
| 201401CP00025|    1|
| 201401CP00212|    1|
| 201401CP00264|    1|
| 201401CP00267|    1|
+--------------+-----+
only showing top 10 rows
```

In [13]:
```
accident.createOrReplaceTempView("accident")
```

In [14]:
```
sqlc.sql("""
         SELECT Accident_Severity, count(*) as count
         FROM accident
         GROUP BY Accident_Severity
""").show(10)
```

```
[Stage 7:>                                                  (0 + 2) / 2]
+-----------------+------+
|Accident_Severity| count|
+-----------------+------+
|           Slight|355591|
|            Fatal|  4969|
|          Serious| 62439|
+-----------------+------+
```

In [15]:
```
sqlc.sql("""
         SELECT Carriageway_Hazards, count(*) as count
         FROM accident
         GROUP BY Carriageway_Hazards
""").show(10)
```

```
+--------------------+------+
| Carriageway_Hazards| count|
+--------------------+------+
|                None|415576|
|   Previous accident|   572|
|Other object on road|  2824|
|Vehicle load on road|   511|
|Pedestrian in car...|  1016|
|Any animal in car...|  1858|
|Data missing or o...|   642|
+--------------------+------+
```

In [16]:
```
sqlc.sql("""
         SELECT Did_Police_Officer_Attend_Scene_of_Accident, count(*) as count
         FROM accident
         GROUP BY Did_Police_Officer_Attend_Scene_of_Accident
         order by Did_Police_Officer_Attend_Scene_of_Accident asc
""").show(10)
```

```
[Stage 13:============================>                            (1 + 1) / 2]
+-----------------------------------------+------+
|Did_Police_Officer_Attend_Scene_of_Accident| count|
+-----------------------------------------+------+
|                                     null|     8|
|                                      1.0|335944|
|                                      2.0| 84864|
|                                      3.0|  2183|
+-----------------------------------------+------+
```

In [17]:
```
sqlc.sql("""
         SELECT Latitude, Longitude, count(*) as count
         FROM accident
         GROUP BY Latitude, Longitude
         order by Latitude asc
""").show(10)
```

```
[Stage 16:============================>                            (1 + 1) / 2]
+---------+---------+-----+
| Latitude|Longitude|count|
+---------+---------+-----+
|     null|     null|   34|
|49.913077|-6.299469|    1|
|49.915252|-6.317417|    1|
|49.915618|-6.311427|    1|
|49.919716|-6.307503|    1|
|49.974574| -5.20813|    1|
|49.975186|-5.204237|    1|
|49.986589|-5.209907|    1|
|49.994462|-5.185075|    1|
|50.005195|-5.249896|    1|
+---------+---------+-----+
only showing top 10 rows
```

In [18]:
```
sqlc.sql("""
         SELECT Light_Conditions, count(*) as count
         FROM accident
         GROUP BY Light_Conditions
         order by Light_Conditions asc
""").show(10)
```

```
+--------------------+------+
|    Light_Conditions| count|
+--------------------+------+
|Darkness - lighti...|  5845|
|Darkness - lights...| 83569|
|Darkness - lights...|  2448|
|Darkness - no lig...| 21476|
|Data missing or o...|    13|
|            Daylight|309648|
+--------------------+------+
```

In [19]:
```
sqlc.sql("""
         SELECT Road_Surface_Conditions, count(*) as count
         FROM accident
         GROUP BY Road_Surface_Conditions
         order by Road_Surface_Conditions asc
""").show(10)
```

```
+----------------------+------+
|Road_Surface_Conditions| count|
+----------------------+------+
|   Data missing or o...|  1275|
|                   Dry|302580|
|   Flood over 3cm. deep|   601|
|           Frost or ice|  5161|
|                  Snow|  1004|
|            Wet or damp|112378|
+----------------------+------+
```

In [20]:
```
sqlc.sql("""
         SELECT Speed_limit, count(*) as count
         FROM accident
         GROUP BY Speed_limit
         order by Speed_limit asc
""").show(10)
```

```
+-----------+------+
|Speed_limit| count|
+-----------+------+
|       null|    37|
|        0.0|     1|
|       10.0|     2|
|       20.0| 14457|
|       30.0|269620|
|       40.0| 35271|
|       50.0| 17110|
|       60.0| 57480|
|       70.0| 29021|
+-----------+------+
```

In [21]:
```
sqlc.sql("""
         SELECT Time, count(*) as count
```

```
        FROM accident
        GROUP BY Time
        order by Time asc
""").show(10)
```

```
[Stage 28:============================>                            (1 + 1) / 2]
+-----+-----+
| Time|count|
+-----+-----+
| null|   20|
|00:01|  510|
|00:02|   80|
|00:03|   68|
|00:04|   62|
|00:05|  301|
|00:06|   63|
|00:07|   54|
|00:08|   57|
|00:09|   53|
+-----+-----+
only showing top 10 rows
```

In [22]:
```
sqlc.sql("""
        SELECT Urban_or_Rural_Area, count(*) as count
        FROM accident
        GROUP BY Urban_or_Rural_Area
        order by Urban_or_Rural_Area asc
""").show(10)
```

```
+------------------+------+
|Urban_or_Rural_Area| count|
+------------------+------+
|             Rural|146157|
|       Unallocated|     7|
|             Urban|276835|
+------------------+------+
```

In [23]:
```
sqlc.sql("""
        SELECT Weather_Conditions, count(*) as count
        FROM accident
        GROUP BY Weather_Conditions
        order by Weather_Conditions asc
""").show(10)
```

```
+--------------------+------+
|  Weather_Conditions| count|
+--------------------+------+
|Data missing or o...|    13|
|   Fine + high winds|  5190|
|  Fine no high winds|345069|
|         Fog or mist|  2160|
|               Other|  6562|
|Raining + high winds|  6084|
|Raining no high w...| 47716|
|Snowing + high winds|   332|
|Snowing no high w...|  1113|
|             Unknown|  8760|
+--------------------+------+
```

In [24]:
```
sqlc.sql("""
         SELECT Year, count(*) as count
         FROM accident
         GROUP BY Year
         order by Year asc
""").show(10)
```

```
+----+------+
|Year| count|
+----+------+
|2014|146322|
|2015|140056|
|2016|136621|
+----+------+
```

Vehicle Data CSV Wrangling

In [25]:
```
vehicle = sqlc.read.csv("/home/ubuntu/notebooks/vehicle_data.csv",header='true')
```

In [26]:
```
!head -n 100000 vehicle_data.csv | csvstat
```

1. "Accident_Index"

Type of data:	Text
Contains null values:	False
Unique values:	83483
Longest value:	13 characters
Most common values:	2004130260804 (8x) 200432B186804 (7x) 2004440WE0968 (7x) 200401TC00843 (6x) 2004030000093 (6x)

2. "Age_Band_of_Driver"

Type of data:	Text
Contains null values:	False
Unique values:	11
Longest value:	28 characters
Most common values:	36 - 45 (22599x) 26 - 35 (22087x) 46 - 55 (14526x) 21 - 25 (10309x) 56 - 65 (8967x)

3. "Age_of_Vehicle"

Type of data:	Number
Contains null values:	True (excluded from calculations)
Unique values:	45
Smallest value:	1
Largest value:	49
Sum:	540884
Mean:	6.496
Median:	6
StDev:	4.653
Most common values:	None (16735x) 1 (10140x) 2 (8666x) 4 (7820x) 3 (7599x)

4. "Driver_Home_Area_Type"

Type of data:	Text
Contains null values:	False
Unique values:	4
Longest value:	28 characters
Most common values:	Urban area (57870x) Data missing or out of range (28237x) Rural (7725x) Small town (6167x)

5. "Driver_IMD_Decile"

Type of data:	Number
Contains null values:	True (excluded from calculations)
Unique values:	11

```
      Smallest value:        1
      Largest value:         10
      Sum:                   372293
      Mean:                  5.381
      Median:                5
      StDev:                 2.855
      Most common values:    None (30809x)
                             1 (7305x)
                             2 (7295x)
                             5 (7184x)
                             3 (7021x)
```

6. "Engine_Capacity_.CC."

```
      Type of data:          Number
      Contains null values:  True (excluded from calculations)
      Unique values:         1151
      Smallest value:        2
      Largest value:         91000
      Sum:                   198799400
      Mean:                  2281.091
      Median:                1698
      StDev:                 2491.437
      Most common values:    None (12848x)
                             1998 (4392x)
                             1997 (2680x)
                             1598 (2537x)
                             1242 (1911x)
```

7. "Hit_Object_in_Carriageway"

```
      Type of data:          Text
      Contains null values:  True (excluded from calculations)
      Unique values:         12
      Longest value:         28 characters
      Most common values:    None (95886x)
                             Kerb (1634x)
                             Parked vehicle (1286x)
                             Other object (374x)
                             Bollard or refuge (372x)
```

8. "Hit_Object_off_Carriageway"

```
      Type of data:          Text
      Contains null values:  True (excluded from calculations)
      Unique values:         12
      Longest value:         29 characters
      Most common values:    None (91619x)
                             Other permanent object (2786x)
                             Tree (1242x)
                             Entered ditch (835x)
                             Near/Offside crash barrier (822x)
```

9. "Journey_Purpose_of_Driver"

```
      Type of data:          Text
      Contains null values:  False
```

```
       Unique values:         1
       Longest value:         28 characters
       Most common values:    Data missing or out of range (99999x)

10. "Junction_Location"

       Type of data:          Text
       Contains null values:  False
       Unique values:         1
       Longest value:         28 characters
       Most common values:    Data missing or out of range (99999x)

11. "make"

       Type of data:          Text
       Contains null values:  False
       Unique values:         248
       Longest value:         18 characters
       Most common values:    VAUXHALL (9719x)
                              FORD (8764x)
                              RENAULT (6187x)
                              HONDA (5603x)
                              PEUGEOT (5437x)

12. "model"

       Type of data:          Text
       Contains null values:  True (excluded from calculations)
       Unique values:         7457
       Longest value:         27 characters
       Most common values:    None (21865x)
                              TRANSIT 350 LWB TD (760x)
                              CLIO DYNAMIQUE 16V (610x)
                              ASTRA LS I (395x)
                              TRANSIT 280 SWB TD (377x)

13. "Propulsion_Code"

       Type of data:          Text
       Contains null values:  True (excluded from calculations)
       Unique values:         8
       Longest value:         16 characters
       Most common values:    Petrol (59681x)
                              Heavy oil (30069x)
                              None (10126x)
                              Gas/Bi-fuel (46x)
                              Petrol/Gas (LPG) (34x)

14. "Sex_of_Driver"

       Type of data:          Text
       Contains null values:  False
       Unique values:         3
       Longest value:         9 characters
       Most common values:    Male (71754x)
                              Female (25464x)
                              Not known (2781x)
```

15. "Skidding_and_Overturning"

Type of data:	Text
Contains null values:	True (excluded from calculations)
Unique values:	7
Longest value:	28 characters
Most common values:	None (84446x) Skidded (12055x) Skidded and overturned (1873x) Overturned (1505x) Jackknifed (77x)

16. "Towing_and_Articulation"

Type of data:	Text
Contains null values:	False
Unique values:	7
Longest value:	28 characters
Most common values:	No tow/articulation (97075x) Articulated vehicle (2222x) Single trailer (474x) Other tow (119x) Caravan (66x)

17. "Vehicle_Leaving_Carriageway"

Type of data:	Text
Contains null values:	False
Unique values:	10
Longest value:	37 characters
Most common values:	Did not leave carriageway (88233x) Nearside (5891x) Offside (3104x) Nearside and rebounded (879x) Offside on to central reservation (545x)

18. "Vehicle_Location.Restricted_Lane"

Type of data:	Number
Contains null values:	True (excluded from calculations)
Unique values:	11
Smallest value:	0
Largest value:	9
Sum:	7864
Mean:	0.079
Median:	0
StDev:	0.739
Most common values:	0 (98512x) 2 (445x) 9 (412x) 6 (250x) 3 (99x)

19. "Vehicle_Manoeuvre"

Type of data:	Text

```
       Contains null values:  False
       Unique values:         19
       Longest value:         35 characters
       Most common values:    Going ahead other (49220x)
                              Turning right (9887x)
                              Waiting to go - held up (8109x)
                              Slowing or stopping (5210x)
                              Going ahead right-hand bend (4530x)
```

20. "Vehicle_Reference"

```
       Type of data:          Number
       Contains null values:  False
       Unique values:         16
       Smallest value:        1
       Largest value:         18
       Sum:                   155648
       Mean:                  1.556
       Median:                1
       StDev:                 0.749
       Most common values:    1 (54916x)
                              2 (37460x)
                              3 (5727x)
                              4 (1333x)
                              5 (343x)
```

21. "Vehicle_Type"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         13
       Longest value:         36 characters
       Most common values:    109 (69635x)
                              106 (6521x)
                              Bus or coach (17 or more pass seats) (6418x)
                              Goods 7.5 tonnes mgw and over (5410x)
                              Van / Goods 3.5 tonnes mgw or under (4977x)
```

22. "Was_Vehicle_Left_Hand_Drive"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         1
       Longest value:         28 characters
       Most common values:    Data missing or out of range (99999x)
```

23. "X1st_Point_of_Impact"

```
       Type of data:          Text
       Contains null values:  False
       Unique values:         6
       Longest value:         28 characters
       Most common values:    Front (50606x)
                              Back (18516x)
                              Offside (13006x)
                              Nearside (11327x)
                              Did not impact (5358x)
```

```
24. "Year"

        Type of data:          Number
        Contains null values:  False
        Unique values:         1
        Smallest value:        2004
        Largest value:         2004
        Sum:                   200397996
        Mean:                  2004
        Median:                2004
        StDev:                 0
        Most common values:    2004 (99999x)

Row count: 99999
```

In [27]:
```
!wc -l vehicle_data.csv
```

```
2177206 vehicle_data.csv
```

In [28]:
```
!csvcut -n vehicle_data.csv
```

```
  1: Accident_Index
  2: Age_Band_of_Driver
  3: Age_of_Vehicle
  4: Driver_Home_Area_Type
  5: Driver_IMD_Decile
  6: Engine_Capacity_.CC.
  7: Hit_Object_in_Carriageway
  8: Hit_Object_off_Carriageway
  9: Journey_Purpose_of_Driver
 10: Junction_Location
 11: make
 12: model
 13: Propulsion_Code
 14: Sex_of_Driver
 15: Skidding_and_Overturning
 16: Towing_and_Articulation
 17: Vehicle_Leaving_Carriageway
 18: Vehicle_Location.Restricted_Lane
 19: Vehicle_Manoeuvre
 20: Vehicle_Reference
 21: Vehicle_Type
 22: Was_Vehicle_Left_Hand_Drive
 23: X1st_Point_of_Impact
 24: Year
```

In [29]:
```
vehicle.groupBy("Accident_Index").count().orderBy("count", ascending=False).show(10)
```

```
[Stage 43:==============================>                          (1 + 1) / 2]
```

```
+--------------+-----+
|Accident_Index|count|
+--------------+-----+
| 2013460234852|   53|
| 201543P296025|   37|
| 2011160B00431|   24|
| 2009559D05333|   22|
| 201522D501706|   16|
| 2016140142191|   16|
| 2015460257544|   14|
| 200911NE16289|   14|
| 201450JC2B008|   13|
| 2016460044087|   13|
+--------------+-----+
only showing top 10 rows
```

In [30]:
```
vehicle.createOrReplaceTempView("vehicle")
```

We are changing the name of the column as below because it will create trouble while cleaning the data

In [31]:
```
vehicle = vehicle.withColumnRenamed("Engine_Capacity_.CC.", "Engine_Capacity")
```

In [32]:
```
vehicle = vehicle.withColumnRenamed("Vehicle_Location.Restricted_Lane", "Vehicle_Locat
```

Checking Nulls/NAs/Data Missing

In [33]:
```
sqlc.sql("""
         SELECT Was_Vehicle_Left_Hand_Drive, count(*) as count
         FROM vehicle
         GROUP BY Was_Vehicle_Left_Hand_Drive
         order by count desc
""").show(10)
```

```
[Stage 44:=============================================>           (4 + 1) / 5]
+--------------------------+-------+
|Was_Vehicle_Left_Hand_Drive|  count|
+--------------------------+-------+
|                        No|2045375|
|       Data missing or o...| 127943|
|                       Yes|   3887|
+--------------------------+-------+
```

We will drop this column as only 3000 of the 2 million rows provide any information.

In [34]:
```
sqlc.sql("""
         SELECT X1st_Point_of_Impact, count(*) as count
         FROM vehicle
         GROUP BY X1st_Point_of_Impact
```

```
        order by count desc
""").show(10)
```

```
[Stage 47:=============================================>           (4 + 1) / 5]
+--------------------+-------+
|X1st_Point_of_Impact|  count|
+--------------------+-------+
|               Front|1060289|
|                Back| 410896|
|             Offside| 305071|
|            Nearside| 273842|
|      Did not impact| 124336|
|Data missing or o...|   2771|
+--------------------+-------+
```

In [35]:
```
sqlc.sql("""
        SELECT Vehicle_Type, count(*) as count
        FROM vehicle
        GROUP BY Vehicle_Type
        order by count desc
""").show(10)
```

```
[Stage 50:=============================================>           (4 + 1) / 5]
+--------------------+-------+
|        Vehicle_Type|  count|
+--------------------+-------+
|                 Car|1528628|
|Van / Goods 3.5 t...| 117427|
|                 109|  82920|
|Bus or coach (17 ...|  76757|
|Motorcycle over 5...|  71472|
|Motorcycle 125cc ...|  61600|
|Goods 7.5 tonnes ...|  55426|
|Taxi/Private hire...|  43781|
|         Pedal cycle|  38904|
|Motorcycle 50cc a...|  22415|
+--------------------+-------+
only showing top 10 rows
```

In [36]:
```
sqlc.sql("""
        SELECT Vehicle_Manoeuvre, count(*) as count
        FROM vehicle
        GROUP BY Vehicle_Manoeuvre
        order by count desc
""").show(10)
```

```
[Stage 53:=============================================>           (4 + 1) / 5]
```

```
+--------------------+------+
|   Vehicle_Manoeuvre| count|
+--------------------+------+
|   Going ahead other|994636|
|       Turning right|216201|
| Slowing or stopping|177548|
|Waiting to go - h...|155643|
|Going ahead right...| 90893|
|              Parked| 88354|
|          Moving off| 87295|
|Going ahead left-...| 78667|
|        Turning left| 70143|
|Overtaking moving...| 43773|
+--------------------+------+
only showing top 10 rows
```

In [37]:
```
sqlc.sql("""
         SELECT Age_Band_of_Driver, count(*) as count
         FROM vehicle
         GROUP BY Age_Band_of_Driver
         order by count desc
""").show(10)
```

```
[Stage 56:=============================================>           (4 + 1) / 5]
+--------------------+------+
|  Age_Band_of_Driver| count|
+--------------------+------+
|             26 - 35|450531|
|             36 - 45|435686|
|             46 - 55|348762|
|             21 - 25|238765|
|             56 - 65|206181|
|             16 - 20|175874|
|Data missing or o...|171052|
|             66 - 75| 91454|
|             Over 75| 54236|
|             11 - 15|  3655|
+--------------------+------+
only showing top 10 rows
```

In [38]:
```
sqlc.sql("""
         SELECT Sex_of_Driver, count(*) as count
         FROM vehicle
         GROUP BY Sex_of_Driver
         order by count desc
""").show(10)
```

```
[Stage 59:=============================================>           (4 + 1) / 5]
```

```
+--------------------+-------+
|       Sex_of_Driver|  count|
+--------------------+-------+
|                Male|1468081|
|              Female| 633005|
|           Not known|  76051|
|Data missing or o...|     68|
+--------------------+-------+
```

In [39]:
```
sqlc.sql("""
         SELECT Journey_Purpose_of_Driver, count(*) as count
         FROM vehicle
         GROUP BY Journey_Purpose_of_Driver
         order by count desc
""").show(10)
```

```
[Stage 62:==============================================>           (4 + 1) / 5]
+------------------------+------+
|Journey_Purpose_of_Driver| count|
+------------------------+------+
|                Not known|834626|
|     Other/Not known (...|555726|
|     Journey as part o...|391713|
|     Commuting to/from...|202525|
|     Data missing or o...|133560|
|                   Other| 32965|
|     Taking pupil to/f...| 21792|
|     Pupil riding to/f...|  4298|
+------------------------+------+
```

We will drop this column as around 75% of the journey purpose data is unknown.

In [40]:
```
sqlc.sql("""
         SELECT Hit_Object_off_Carriageway, count(*) as count
         FROM vehicle
         GROUP BY Hit_Object_off_Carriageway
         order by count desc
""").show(10)
```

```
[Stage 65:==============================================>           (4 + 1) / 5]
```

```
+-------------------------+-------+
|Hit_Object_off_Carriageway|  count|
+-------------------------+-------+
|                     None|1989741|
|      Other permanent o...|  56640|
|                     Tree|  29161|
|            Entered ditch|  18301|
|      Road sign or traf...|  17394|
|      Central crash bar...|  16538|
|      Near/Offside cras...|  15835|
|                Lamp post|  15293|
|            Wall or fence|  10423|
|      Telegraph or elec...|   5828|
+-------------------------+-------+
only showing top 10 rows
```

In [41]:
```
sqlc.sql("""
         SELECT Hit_Object_in_Carriageway, count(*) as count
         FROM vehicle
         GROUP BY Hit_Object_in_Carriageway
         order by count desc
""").show(10)
```

```
[Stage 68:=============================================>           (4 + 1) / 5]
+------------------------+-------+
|Hit_Object_in_Carriageway|  count|
+------------------------+-------+
|                    None|2087824|
|                    Kerb|  34354|
|           Parked vehicle|  27401|
|        Bollard or refuge|   9981|
|             Other object|   5715|
|     Any animal (excep...|   2683|
|     Central island of...|   2190|
|     Open door of vehicle|   1919|
|            Bridge (side)|   1441|
|     Data missing or o...|   1302|
+------------------------+-------+
only showing top 10 rows
```

In [42]:
```
sqlc.sql("""
         SELECT Driver_Home_Area_Type, count(*) as count
         FROM vehicle
         GROUP BY Driver_Home_Area_Type
         order by count desc
""").show(10)
```

```
[Stage 71:=============================================>           (4 + 1) / 5]
```

```
+---------------------+-------+
|Driver_Home_Area_Type|  count|
+---------------------+-------+
|           Urban area|1436598|
| Data missing or o...| 334344|
|                Rural| 232360|
|           Small town| 173903|
+---------------------+-------+
```

In [43]:
```
sqlc.sql("""
         SELECT Year, count(*) as count
         FROM vehicle
         GROUP BY Year
         order by count desc
""").show(10)
```

```
[Stage 74:=============================================>           (4 + 1) / 5]
+----+------+
|Year| count|
+----+------+
|2015|257845|
|2016|252500|
|2014|182353|
|2009|182321|
|2011|180616|
|2010|180367|
|2012|173859|
|2013|171625|
|2007|127172|
|2008|122445|
+----+------+
only showing top 10 rows
```

We will drop this column as this matches with the year in the Accident data table.

In [44]:
```
sqlc.sql("""
         SELECT Age_of_Vehicle, count(*) as count
         FROM vehicle
         GROUP BY Age_of_Vehicle
         order by count desc
""").show(10)
```

```
[Stage 77:=============================================>           (4 + 1) / 5]
```

```
+--------------+------+
|Age_of_Vehicle| count|
+--------------+------+
|            NA|358149|
|             1|180333|
|             2|161072|
|             3|148665|
|             4|144493|
|             5|138464|
|             6|134524|
|             7|130114|
|             8|127441|
|             9|121279|
+--------------+------+
only showing top 10 rows
```

In [45]:
```
sqlc.sql("""
         SELECT model, count(*) as count
         FROM vehicle
         GROUP BY model
         order by count desc
""").show(10)
```

```
[Stage 80:=============================================>           (4 + 1) / 5]
+------------------+------+
|             model| count|
+------------------+------+
|              null|214486|
|              NULL|110845|
|           MISSING| 11877|
|CLIO DYNAMIQUE 16V|  8193|
|      PUNTO ACTIVE|  5348|
|TRANSIT 350 LWB TD|  4529|
|                KA|  4489|
|            206 LX|  4253|
|   PUNTO ACTIVE 8V|  4115|
|  SPRINTER 313 CDI|  3773|
+------------------+------+
only showing top 10 rows
```

In [46]:
```
sqlc.sql("""
         SELECT make, count(*) as count
         FROM vehicle
         GROUP BY make
         order by count desc
""").show(10)
```

```
[Stage 83:=============================================>           (4 + 1) / 5]
```

```
+----------+------+
|      make| count|
+----------+------+
|  VAUXHALL|239650|
|      FORD|237084|
|   PEUGEOT|126533|
|VOLKSWAGEN|120820|
|   RENAULT|114300|
|      NULL|110845|
|     HONDA|103165|
|  MERCEDES| 89891|
|    TOYOTA| 83597|
|   CITROEN| 81787|
+----------+------+
only showing top 10 rows
```

In [47]:
```
sqlc.sql("""
         SELECT Driver_IMD_Decile, count(*) as count
         FROM vehicle
         GROUP BY Driver_IMD_Decile
         order by count desc
""").show(10)
```

```
[Stage 86:=============================================>           (4 + 1) / 5]
+-----------------+------+
|Driver_IMD_Decile| count|
+-----------------+------+
|               NA|734812|
|                3|151377|
|                4|151068|
|                2|151065|
|                5|150367|
|                6|149351|
|                7|144579|
|                1|141939|
|                8|140578|
|                9|136802|
+-----------------+------+
only showing top 10 rows
```

We will drop this table as it's irrelevant to our analytical questions.

In [48]:
```
sqlc.sql("""
         SELECT Junction_Location, count(*) as count
         FROM vehicle
         GROUP BY Junction_Location
         order by count desc
""").show(10)
```

```
[Stage 89:=============================================>           (4 + 1) / 5]
```

```
+--------------------+------+
|   Junction_Location| count|
+--------------------+------+
|Not at or within ...|808108|
|Approaching junct...|476804|
|Mid Junction - on...|423304|
|Data missing or o...|122612|
|Cleared junction ...|114061|
|  Entering main road| 89797|
| Entering roundabout| 54546|
|   Leaving main road| 51094|
|  Leaving roundabout| 28965|
|Entering from sli...|  7914|
+--------------------+------+
```

In [49]:
```
sqlc.sql("""
         SELECT Propulsion_Code, count(*) as count
         FROM vehicle
         GROUP BY Propulsion_Code
         order by count desc
""").show(10)
```

```
[Stage 92:=============================================>           (4 + 1) / 5]
+-------------------+-------+
|    Propulsion_Code|  count|
+-------------------+-------+
|             Petrol|1143097|
|          Heavy oil| 775829|
|                 NA| 245843|
|    Hybrid electric|   9172|
|        Gas/Bi-fuel|   1642|
|   Petrol/Gas (LPG)|    664|
|           Electric|    618|
|                Gas|    175|
|    Electric diesel|    143|
|New fuel technology|     11|
+-------------------+-------+
only showing top 10 rows
```

We will drop this column because the petrol information of the vehicle is not relevant to our analysis.

In [50]:
```
sqlc.sql("""
         SELECT Skidding_and_Overturning, count(*) as count
         FROM vehicle
         GROUP BY Skidding_and_Overturning
         order by count desc
""").show(10)
```

```
[Stage 95:=============================================>           (4 + 1) / 5]
```

```
+-----------------------+-------+
|Skidding_and_Overturning|  count|
+-----------------------+-------+
|                   None|1898208|
|                Skidded| 199825|
|    Skidded and overt...|  42339|
|             Overturned|  34100|
|             Jackknifed|   1127|
|    Data missing or o...|   1015|
|    Jackknifed and ov...|    591|
+-----------------------+-------+
```

In [51]:
```
sqlc.sql("""
         SELECT Towing_and_Articulation, count(*) as count
         FROM vehicle
         GROUP BY Towing_and_Articulation
         order by count desc
""").show(10)
```

```
[Stage 98:=============================================>           (4 + 1) / 5]
+----------------------+-------+
|Towing_and_Articulation|  count|
+----------------------+-------+
|    No tow/articulation|2136394|
|    Articulated vehicle|  26930|
|         Single trailer|   8855|
|              Other tow|   2002|
|                Caravan|   1332|
|   Data missing or o...|   1176|
|   Double or multipl...|    516|
+----------------------+-------+
```

We will drop this column as it's not relevant to our analysis.

In [52]:
```
sqlc.sql("""
         SELECT Vehicle_Leaving_Carriageway, count(*) as count
         FROM vehicle
         GROUP BY Vehicle_Leaving_Carriageway
         order by count desc
""").show(10)
```

```
[Stage 101:=============================================>           (4 + 1) / 5]
```

```
+--------------------------+-------+
|Vehicle_Leaving_Carriageway|  count|
+--------------------------+-------+
|       Did not leave car...|1922892|
|                   Nearside| 131169|
|                    Offside|  67104|
|       Nearside and rebo...|  17237|
|       Offside on to cen...|  10798|
|       Offside and rebou...|   8863|
|       Straight ahead at...|   8096|
|       Offside on to cen...|   7677|
|       Offside - crossed...|   2151|
|       Data missing or o...|   1218|
+--------------------------+-------+
```

In [53]:
```
sqlc.sql("""
         SELECT Vehicle_Reference, count(*) as count
         FROM vehicle
         GROUP BY Vehicle_Reference
         order by count desc
""").show(10)
```

```
[Stage 104:=============================================>           (4 + 1) / 5]
+-----------------+-------+
|Vehicle_Reference|  count|
+-----------------+-------+
|                1|1202113|
|                2| 809191|
|                3| 125564|
|                4|  28518|
|                5|   7221|
|                6|   2461|
|                7|    995|
|                8|    469|
|                9|    239|
|               10|    133|
+-----------------+-------+
only showing top 10 rows
```

Dropping columns with high number of nulls/information that will not be used in analysis as it doesn't tell us anything unique

In [54]:
```
!csvcut -c 1,2,3,4,7,8,10,11,12,14,15,17,19,20,21,23 vehicle_data.csv > vehicle_update
```

```
Your file is not "utf-8-sig" encoded. Please specify the correct encoding with the -e
flag or with the PYTHONIOENCODING environment variable. Use the -v flag to see the co
mplete error.
```

In [55]:
```
!csvcut -n vehicle_data.csv
```

```
  1: Accident_Index
  2: Age_Band_of_Driver
  3: Age_of_Vehicle
  4: Driver_Home_Area_Type
  5: Driver_IMD_Decile
  6: Engine_Capacity_.CC.
  7: Hit_Object_in_Carriageway
  8: Hit_Object_off_Carriageway
  9: Journey_Purpose_of_Driver
 10: Junction_Location
 11: make
 12: model
 13: Propulsion_Code
 14: Sex_of_Driver
 15: Skidding_and_Overturning
 16: Towing_and_Articulation
 17: Vehicle_Leaving_Carriageway
 18: Vehicle_Location.Restricted_Lane
 19: Vehicle_Manoeuvre
 20: Vehicle_Reference
 21: Vehicle_Type
 22: Was_Vehicle_Left_Hand_Drive
 23: X1st_Point_of_Impact
 24: Year
```

In [56]:
```
!csvcut -n vehicle_updated.csv
```

```
  1: Accident_Index
  2: Age_Band_of_Driver
  3: Age_of_Vehicle
  4: Driver_Home_Area_Type
  5: Hit_Object_in_Carriageway
  6: Hit_Object_off_Carriageway
  7: Junction_Location
  8: make
  9: model
 10: Sex_of_Driver
 11: Skidding_and_Overturning
 12: Vehicle_Leaving_Carriageway
 13: Vehicle_Manoeuvre
 14: Vehicle_Reference
 15: Vehicle_Type
 16: X1st_Point_of_Impact
```

Creating Dimensional Model

Create Accident table

In [57]:
```
!dropdb -U student final_project_10
```

In [58]:
```
!createdb -U student final_project_10
```

In [59]:
```
%sql postgresql://student@/final_project_10
```

In [60]:
```
!pwd
```

```
/home/ubuntu/notebooks
```

In [61]:
```
%%sql
DROP TABLE IF EXISTS Accident;

CREATE TABLE Accident (
 Accident_Index VARCHAR NOT NULL,
 Accident_Severity VARCHAR(100),
 Carriageway_Hazards VARCHAR(50),
 Date Date,
 Day_of_Week Varchar(20),
 Did_Police_Officer_Attend_Scene_of_Accident NUMERIC(5),
 Latitude FLOAT(20),
 Light_Conditions Varchar(100),
 Local_Authority_District Varchar(100),
 Longitude FLOAT(20),
 Number_of_Casualties Numeric(10),
 Number_of_Vehicles Numeric(10),
 Police_Force VARCHAR(100),
 Road_Surface_Conditions VARCHAR(100),
 Speed_limit NUMERIC(3),
 Time Varchar,
 Urban_or_Rural_Area Varchar(20),
 Weather_Conditions Varchar(100),
 Year Numeric(5),
 Age_Band_of_Driver VARCHAR(100),
 Age_of_Vehicle NUMERIC,
 Driver_Home_Area_Type VARCHAR(100),
 Hit_Object_in_Carriageway VARCHAR(100),
 Hit_Object_off_Carriageway VARCHAR(100),
 Junction_Location VARCHAR(100),
 make VARCHAR(100),
 model VARCHAR(100),
 Sex_of_Driver VARCHAR(100),
 Skidding_and_Overturning VARCHAR(100),
 Vehicle_Leaving_Carriageway VARCHAR(100),
 Vehicle_Manoeuvre VARCHAR(100),
 Vehicle_Reference NUMERIC,
 Vehicle_Type VARCHAR(100),
 X1st_Point_of_Impact VARCHAR(100)
 );
```

```
 * postgresql://student@/final_project_10
Done.
Done.
```

Out[61]: []

In [62]:
```
%%sql
select * from Accident
```

```
 * postgresql://student@/final_project_10
0 rows affected.
```

Out[62]:

accident_index	accident_severity	carriageway_hazards	date	day_of_week	did_police_officer_attend_sce

In [63]:
```
%%sql
COPY Accident FROM '/home/ubuntu/notebooks/accident_final.csv'
CSV
HEADER;
```

```
 * postgresql://student@/final_project_10
692698 rows affected.
```

Out[63]: []

In [64]:
```
%%sql
select * from Accident
limit 5;
```

```
 * postgresql://student@/final_project_10
5 rows affected.
```

Out[64]:

accident_index	accident_severity	carriageway_hazards	date	day_of_week	did_police_officer_attend_s
201401BS70001	Slight	None	2014-01-09	Thursday	
201401BS70001	Slight	None	2014-01-09	Thursday	
201401BS70002	Slight	None	2014-01-20	Monday	
201401BS70003	Slight	None	2014-01-21	Tuesday	
201401BS70003	Slight	None	2014-01-21	Tuesday	

Converting the NULLS to 9999 or "Data Missing"

In [65]:
```
%%sql
update Accident
set Carriageway_Hazards = 'No object present'
where Carriageway_Hazards = 'None';
update Accident
set Carriageway_Hazards = 'DATA MISSING'
where Carriageway_Hazards = 'Data missing or out of range';
update Accident
set Latitude = 0
where Latitude is null;
update Accident
set Longitude = 0
where Longitude is null;
```

```
update Accident
set Did_Police_Officer_Attend_Scene_of_Accident = 0
where Did_Police_Officer_Attend_Scene_of_Accident is null;
update Accident
set Light_Conditions = 'DATA MISSING'
where Light_Conditions = 'Data missing or out of range';
update Accident
set Road_Surface_Conditions = 'DATA MISSING'
where Road_Surface_Conditions = 'Data missing or out of range';
update Accident
set Speed_limit = 99
where Speed_limit is null;
update Accident
set Urban_or_Rural_Area = 'DATA MISSING'
where Urban_or_Rural_Area = 'Unallocated';
update Accident
set Weather_Conditions = 'DATA MISSING'
where Weather_Conditions = 'Data missing or out of range' or Weather_Conditions = 'Unk
update Accident
set time = 'DATA MISSING'
where time is null;
```

```
 * postgresql://student@/final_project_10
681344 rows affected.
1078 rows affected.
65 rows affected.
65 rows affected.
16 rows affected.
25 rows affected.
2033 rows affected.
65 rows affected.
12 rows affected.
13733 rows affected.
37 rows affected.
```

Out[65]: []

Create the Accident _Type table as a dimension table

In [66]:
```
%%sql
DROP TABLE IF EXISTS Accident_Type Cascade;
CREATE TABLE Accident_Type (
 Key SERIAL PRIMARY KEY,
 Accident_Severity VARCHAR(100),
 Number_of_Casualties Numeric(10),
 Number_of_Vehicles Numeric(10),
 Did_Police_Officer_Attend_Scene_of_Accident NUMERIC(5)
 );
```

```
 * postgresql://student@/final_project_10
Done.
Done.
```

Out[66]: []

Populate the Accident_Type table with data from table Accident

In [67]:
```
%%sql
INSERT INTO Accident_Type (Accident_Severity, Number_of_Casualties, Number_of_Vehicles
SELECT DISTINCT Accident_Severity, Number_of_Casualties, Number_of_Vehicles, Did_Polic
Accident;
```

```
 * postgresql://student@/final_project_10
429 rows affected.
```

Out[67]: []

In [68]:
```
%%sql
select * from Accident_Type limit 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[68]:

key	accident_severity	number_of_casualties	number_of_vehicles	did_police_officer_attend_scene_of_acci
1	Slight	1	1	
2	Slight	16	2	
3	Slight	21	1	
4	Slight	5	10	
5	Slight	5	2	
6	Slight	14	3	
7	Serious	4	4	
8	Slight	2	10	
9	Serious	2	4	
10	Serious	54	1	

In [69]:
```
%%sql
ALTER TABLE Accident
ADD COLUMN Accident_type_key INTEGER,
ADD CONSTRAINT fk_Accident_type
FOREIGN KEY (Accident_type_key) REFERENCES Accident_type (key);
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[69]: []

In [70]:
```
%%sql
UPDATE Accident AS a
SET Accident_type_key = b.key
FROM Accident_type AS b
WHERE a.Accident_Severity = b.Accident_Severity
and a.Number_of_Casualties= b.Number_of_Casualties
and a.Number_of_Vehicles=b.Number_of_Vehicles
and a.Did_Police_Officer_Attend_Scene_of_Accident=b.Did_Police_Officer_Attend_Scene_of
--and a.Accident_Severity is not null
--and a.Number_of_Casualties is not null
```

```
--and a.Number_of_Vehicles is not null
--and a.Did_Police_Officer_Attend_Scene_of_Accident is not null;
```

```
 * postgresql://student@/final_project_10
692698 rows affected.
```

Out[70]: []

In [71]:
```
%%sql
Select * from Accident where Accident_type_key is NULL;
```

```
 * postgresql://student@/final_project_10
0 rows affected.
```

Out[71]:

accident_index	accident_severity	carriageway_hazards	date	day_of_week	did_police_officer_attend_sce

Create the Conditions table as a dimension table

In [72]:
```
%%sql
DROP TABLE IF EXISTS Conditions;
CREATE TABLE Conditions (
 Key SERIAL PRIMARY KEY,
 Weather_Conditions VARCHAR(100),
 Light_Conditions VARCHAR(100),
 Carriageway_Hazards VARCHAR(50),
 Road_Surface_Conditions VARCHAR(100),
 Speed_limit NUMERIC(3)
 );
```

```
 * postgresql://student@/final_project_10
Done.
Done.
```

Out[72]: []

In [73]:
```
%%sql
INSERT INTO Conditions (Weather_Conditions, Light_Conditions,Carriageway_Hazards, Roac
SELECT DISTINCT Weather_Conditions, Light_Conditions,Carriageway_Hazards, Road_Surface
Accident;
```

```
 * postgresql://student@/final_project_10
1544 rows affected.
```

Out[73]: []

In [74]:
```
%%sql
select * from Conditions limit 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[74]:

key	weather_conditions	light_conditions	carriageway_hazards	road_surface_conditions	speed_limit
1	DATA MISSING	DATA MISSING	DATA MISSING	DATA MISSING	60
2	DATA MISSING	DATA MISSING	No object present	DATA MISSING	30
3	DATA MISSING	DATA MISSING	No object present	DATA MISSING	40
4	DATA MISSING	DATA MISSING	No object present	DATA MISSING	60
5	DATA MISSING	DATA MISSING	No object present	DATA MISSING	99
6	DATA MISSING	Darkness - lighting unknown	Any animal in carriageway (except ridden horse)	Dry	30
7	DATA MISSING	Darkness - lighting unknown	Any animal in carriageway (except ridden horse)	Wet or damp	60
8	DATA MISSING	Darkness - lighting unknown	DATA MISSING	DATA MISSING	20
9	DATA MISSING	Darkness - lighting unknown	DATA MISSING	DATA MISSING	30

In [75]:
```
%%sql
ALTER TABLE Accident
ADD COLUMN Conditions_key INTEGER,
ADD CONSTRAINT fk_Conditions
FOREIGN KEY (Conditions_key) REFERENCES Conditions(key);
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[75]: []

In [76]:
```
%%sql
UPDATE Accident AS a
SET Conditions_key = c.key
FROM Conditions AS c
WHERE a.Weather_Conditions = c.Weather_Conditions
and a.Light_Conditions= c.Light_Conditions
and a.Carriageway_Hazards=c.Carriageway_Hazards
and a.Road_Surface_Conditions=c.Road_Surface_Conditions
and a.Speed_limit=c.Speed_limit
--and a.Weather_Conditions is not null
--and a.Light_Conditions is not null
--and a.Carriageway_Hazards is not null
--and a.Road_Surface_Conditions is not null
--and a.Speed_limit is not null;
```

```
 * postgresql://student@/final_project_10
692698 rows affected.
```

Out[76]: []

In [77]:
```
%%sql
Select * from Conditions limit 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[77]:

key	weather_conditions	light_conditions	carriageway_hazards	road_surface_conditions	speed_limit
1	DATA MISSING	DATA MISSING	DATA MISSING	DATA MISSING	60
2	DATA MISSING	DATA MISSING	No object present	DATA MISSING	30
3	DATA MISSING	DATA MISSING	No object present	DATA MISSING	40
4	DATA MISSING	DATA MISSING	No object present	DATA MISSING	60
5	DATA MISSING	DATA MISSING	No object present	DATA MISSING	99
6	DATA MISSING	Darkness - lighting unknown	Any animal in carriageway (except ridden horse)	Dry	30
7	DATA MISSING	Darkness - lighting unknown	Any animal in carriageway (except ridden horse)	Wet or damp	60
8	DATA MISSING	Darkness - lighting unknown	DATA MISSING	DATA MISSING	20
9	DATA MISSING	Darkness - lighting unknown	DATA MISSING	DATA MISSING	30
10	DATA MISSING	Darkness - lighting unknown	DATA MISSING	DATA MISSING	50

Create the Location table as a dimension table

In [78]:
```
%%sql
DROP TABLE IF EXISTS Location Cascade;
CREATE TABLE Location (
 Key SERIAL PRIMARY KEY,
 Local_Authority_District VARCHAR(100),
 Urban_or_Rural_Area VARCHAR(20),
 Latitude FLOAT(20),
 Longitude FLOAT(20),
 Police_Force VARCHAR(100)
 );
```

```
 * postgresql://student@/final_project_10
Done.
Done.
```

Out[78]: []

In [79]:
```
%%sql
INSERT INTO Location (Local_Authority_District, Urban_or_Rural_Area,Latitude,Longitude
SELECT DISTINCT Local_Authority_District,Urban_or_Rural_Area,Latitude,Longitude,Police
Accident;
```

```
 * postgresql://student@/final_project_10
383998 rows affected.
```

Out[79]: []

In [80]:
```
%%sql
select * from Location limit 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[80]:

key	local_authority_district	urban_or_rural_area	latitude	longitude	police_force
1	Aberdeen City	Rural	56.989235	-2.249393	Grampian
2	Aberdeen City	Rural	57.085915	-2.093664	Grampian
3	Aberdeen City	Rural	57.090897	-2.11592	Grampian
4	Aberdeen City	Rural	57.093872	-2.105929	Grampian
5	Aberdeen City	Rural	57.095142	-2.116099	Grampian
6	Aberdeen City	Rural	57.09521	-2.114201	Grampian
7	Aberdeen City	Rural	57.09581	-2.113064	Grampian
8	Aberdeen City	Rural	57.095875	-2.114418	Grampian
9	Aberdeen City	Rural	57.096493	-2.271552	Grampian
10	Aberdeen City	Rural	57.099415	-2.277403	Grampian

In [81]:
```
%%sql
ALTER TABLE Accident
ADD COLUMN Location_key INTEGER,
ADD CONSTRAINT fk_Location
FOREIGN KEY (Location_key) REFERENCES Location(key);
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[81]: []

In [82]:
```
%%sql
UPDATE Accident AS a
SET Location_key = l.key
FROM Location AS l
WHERE a.Local_Authority_District = l.Local_Authority_District
and a.Urban_or_Rural_Area= l.Urban_or_Rural_Area
and a.Latitude = l.Latitude
and a.Longitude = l.Longitude
and a.Police_Force = l.Police_Force
--and a.Local_Authority_District is not null
--and a.Urban_or_Rural_Area is not null
--and a.Latitude is not null
```

```
--and a.Longitude is not null
--and a.Police_Force is not null;
```

```
 * postgresql://student@/final_project_10
692698 rows affected.
```

Out[82]: []

Create the Time table as a dimension table

In [83]:
```
%%sql
DROP TABLE IF EXISTS Day_time Cascade;

CREATE TABLE Day_time(
    key SERIAL PRIMARY KEY,
    date date,
    Time Varchar,
    year INTEGER,
    month_of_year_str VARCHAR(12),
    month_of_year INTEGER,
    day_of_month INTEGER,
    day_of_week_str CHAR(9),
    day_of_week INTEGER,
    is_weekend VARCHAR,
    is_weekday VARCHAR,
    quarter_of_year INTEGER
);
```

```
 * postgresql://student@/final_project_10
Done.
Done.
```

Out[83]: []

In [84]:
```
%%sql
INSERT INTO Day_time (date,time, year, month_of_year_str, month_of_year, day_of_month,
                  day_of_week_str, day_of_week, is_weekend, is_weekday,
                  quarter_of_year)
SELECT DISTINCT date,
    time,
    CAST(TO_CHAR(date, 'YYYY') AS INTEGER) AS year,
    TO_CHAR(date, 'Month') AS month_of_year_str,
    CAST(TO_CHAR(date, 'MM') AS INTEGER) AS month_of_year,
    CAST(TO_CHAR(date, 'DD') AS INTEGER) AS day_of_month,
    TO_CHAR(date, 'Day') AS day_of_week_str,
    CAST(TO_CHAR(date, 'D') AS INTEGER) AS day_of_week,
    CASE WHEN CAST(TO_CHAR(date, 'D') AS INTEGER) in (7,1)
        THEN 'true'
        ELSE 'false'
    END AS is_weekend,
    CASE WHEN CAST(TO_CHAR(date, 'D') AS INTEGER) Not in (7,1)
        THEN 'true'
        ELSE 'false'
    END AS is_weekday,
    CAST(TO_CHAR(date, 'Q') AS INTEGER) AS quarter_of_year
FROM Accident
```

```
 * postgresql://student@/final_project_10
271120 rows affected.
```

Out[84]: []

In [85]:
```
%%sql
SELECT * FROM Day_time
LIMIT 10;
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[85]:

key	date	time	year	month_of_year_str	month_of_year	day_of_month	day_of_week_str	day_of_we
1	2016-10-19	05:41	2016	October	10	19	Wednesday	
2	2015-11-08	16:45	2015	November	11	8	Sunday	
3	2014-03-14	08:04	2014	March	3	14	Friday	
4	2015-10-03	16:45	2015	October	10	3	Saturday	
5	2015-05-12	06:15	2015	May	5	12	Tuesday	
6	2016-03-24	15:25	2016	March	3	24	Thursday	
7	2015-09-24	20:32	2015	September	9	24	Thursday	
8	2015-02-11	19:27	2015	February	2	11	Wednesday	
9	2015-04-14	13:30	2015	April	4	14	Tuesday	
10	2016-05-19	23:52	2016	May	5	19	Thursday	

In [86]:
```
%%sql
ALTER TABLE Accident
ADD COLUMN Time_key INTEGER,
ADD CONSTRAINT fk_Time
FOREIGN KEY (Time_key) REFERENCES Day_time(key);
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[86]: []

In [87]:
```
%%sql
UPDATE Accident AS a
SET Time_key = t.key
FROM Day_time AS t
```

```
WHERE a.Date = t.Date
and a.Time= t.Time
```

```
 * postgresql://student@/final_project_10
692698 rows affected.
```

Out[87]: []

In [88]:
```
%%sql
select * from Accident limit 10;
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[88]:

accident_index	accident_severity	carriageway_hazards	date	day_of_week	did_police_officer_attend_s
201401BS70126	Slight	Other object on road	2014-03-18	Tuesday	
201401BS70327	Slight	Other object on road	2014-05-20	Tuesday	
201401BS70361	Slight	Pedestrian in carriageway - not injured	2014-06-03	Tuesday	
201401BS70416	Serious	Other object on road	2014-06-26	Thursday	
201401BS70479	Slight	Pedestrian in carriageway - not injured	2014-07-25	Friday	
201401BS70479	Slight	Pedestrian in carriageway - not injured	2014-07-25	Friday	
201401BS70487	Serious	Pedestrian in carriageway - not injured	2014-07-29	Tuesday	
201401BS70518	Slight	Pedestrian in carriageway - not injured	2014-08-08	Friday	
201401CP00019	Slight	Vehicle load on road	2014-01-28	Tuesday	
201401CP00019	Slight	Vehicle load on road	2014-01-28	Tuesday	

In [89]:
```
%%sql
ALTER TABLE Accident
DROP COLUMN accident_severity,
DROP COLUMN carriageway_hazards,
DROP COLUMN date,
DROP COLUMN day_of_week,
DROP COLUMN did_police_officer_attend_scene_of_accident,
DROP COLUMN latitude,
DROP COLUMN light_conditions,
DROP COLUMN local_authority_district,
DROP COLUMN longitude,
DROP COLUMN number_of_casualties,
DROP COLUMN number_of_vehicles,
DROP COLUMN police_force,
DROP COLUMN road_surface_conditions,
DROP COLUMN speed_limit,
DROP COLUMN time,
DROP COLUMN urban_or_rural_area,
DROP COLUMN weather_conditions,
DROP COLUMN year;
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[89]:
```
[]
```

In [90]:
```
%%sql
select * from Accident limit 10;
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[90]:

accident_index	age_band_of_driver	age_of_vehicle	driver_home_area_type	hit_object_in_carriageway	h
201401CP00019	Data missing or out of range	None	Small town	None	
201401CP00019	26 - 35	None	Urban area	None	
201401CP00033	36 - 45	1.0	Urban area	None	
201401CP00057	46 - 55	2.0	Urban area	None	
201401CP00059	36 - 45	5.0	Urban area	None	
201401CP00213	46 - 55	3.0	Urban area	None	
201401CP00307	46 - 55	2.0	Urban area	None	
201401CP00294	36 - 45	7.0	Data missing or out of range	None	
201401CP00388	56 - 65	1.0	Urban area	None	

Create driver detail table as a dimension table

In [91]:
```
%%sql
DROP TABLE IF EXISTS driver CASCADE;

CREATE TABLE driver (
    key SERIAL PRIMARY KEY,
    Age_Band_of_Driver VARCHAR(100),
    Sex_of_Driver VARCHAR(100),
```

```
    Driver_Home_Area_Type VARCHAR(100)
);
```

```
 * postgresql://student@/final_project_10
Done.
Done.
```

Out[91]: []

In [92]:
```
%%sql
INSERT INTO driver (Age_Band_of_Driver, Sex_of_Driver,Driver_Home_Area_Type)
SELECT DISTINCT Age_Band_of_Driver, Sex_of_Driver,Driver_Home_Area_Type
FROM accident;
```

```
 * postgresql://student@/final_project_10
144 rows affected.
```

Out[92]: []

In [93]:
```
%%sql
ALTER TABLE accident
ADD COLUMN driver_key INTEGER,
ADD CONSTRAINT fk_driver_key
    FOREIGN KEY (driver_key)
    REFERENCES driver (key);
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[93]: []

In [94]:
```
%%sql
UPDATE accident
SET driver_key = driver.key
FROM driver
WHERE (accident.age_band_of_driver = driver.age_band_of_driver AND
       accident.sex_of_driver = driver.sex_of_driver AND
       accident.driver_home_area_type = driver.driver_home_area_type);
```

```
 * postgresql://student@/final_project_10
692698 rows affected.
```

Out[94]: []

In [95]:
```
%%sql
SELECT * FROM driver
WHERE age_band_of_driver = '26 - 35' and sex_of_driver = 'Female';
```

```
 * postgresql://student@/final_project_10
4 rows affected.
```

Out[95]:

key	age_band_of_driver	sex_of_driver	driver_home_area_type
46	26 - 35	Female	Data missing or out of range
47	26 - 35	Female	Rural
48	26 - 35	Female	Small town
49	26 - 35	Female	Urban area

In [96]:
```
%%sql
SELECT accident_index, Age_Band_of_Driver, Sex_of_Driver,Driver_Home_Area_Type, driver
FROM accident
WHERE age_band_of_driver = '26 - 35' and sex_of_driver = 'Female'
LIMIT 5;
```

```
 * postgresql://student@/final_project_10
5 rows affected.
```

Out[96]:

accident_index	age_band_of_driver	sex_of_driver	driver_home_area_type	driver_key
201401EK40926	26 - 35	Female	Urban area	49
201401EO40733	26 - 35	Female	Urban area	49
201401JI40283	26 - 35	Female	Urban area	49
201401JI40330	26 - 35	Female	Urban area	49
201401JI40330	26 - 35	Female	Urban area	49

Create vehicle detail table as a dimension table

In [97]:
```
%%sql

DROP TABLE IF EXISTS vehicle_details;

CREATE TABLE vehicle_details (
    key SERIAL PRIMARY KEY,
    make VARCHAR(100),
    model VARCHAR(100),
    Vehicle_Type VARCHAR(100),
    Age_of_Vehicle NUMERIC
);
```

```
 * postgresql://student@/final_project_10
Done.
Done.
```

Out[97]: []

In [98]:
```
%%sql
INSERT INTO vehicle_details (make,model,Vehicle_Type,Age_of_Vehicle)
SELECT DISTINCT make,model,Vehicle_Type,Age_of_Vehicle
FROM accident;
```

```
 * postgresql://student@/final_project_10
104925 rows affected.
```

Out[98]: []

In [99]:
```
%%sql
ALTER TABLE accident
ADD COLUMN vehicle_details_key INTEGER,
ADD CONSTRAINT fk_vehicle_details_key
    FOREIGN KEY (vehicle_details_key)
    REFERENCES vehicle_details (key);
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[99]: []

In [100…
```
%%sql
SELECT * FROM accident
LIMIT 5;
```

```
 * postgresql://student@/final_project_10
5 rows affected.
```

Out[100]:

accident_index	age_band_of_driver	age_of_vehicle	driver_home_area_type	hit_object_in_carriageway	h
201401JI40124	46 - 55	2.0	Urban area	Previous accident	
201401JI40124	21 - 25	10.0	Data missing or out of range	None	
201401JI40124	36 - 45	7.0	Urban area	None	
201401JI40162	26 - 35	11.0	Urban area	None	
201401JI40245	46 - 55	14.0	Urban area	None	

In [101…
```
%%sql
UPDATE accident
SET vehicle_details_key = vehicle_details.key
FROM vehicle_details
WHERE (accident.make = vehicle_details.make AND accident.model = vehicle_details.model
```

```
 * postgresql://student@/final_project_10
500849 rows affected.
```

Out[101]: []

Create accident detail table as a dimension table

In [102…
```
%%sql

DROP TABLE IF EXISTS accident_details;

CREATE TABLE accident_details (
    key SERIAL PRIMARY KEY,
    Hit_Object_in_Carriageway VARCHAR(100),
    Hit_Object_off_Carriageway VARCHAR(100),
    Junction_Location VARCHAR(100),
```

```
    Skidding_and_Overturning VARCHAR(100),
    Vehicle_Leaving_Carriageway VARCHAR(100),
    Vehicle_Manoeuvre VARCHAR(100),
    Vehicle_Reference NUMERIC,
    X1st_Point_of_Impact VARCHAR(100)
);
```

```
 * postgresql://student@/final_project_10
Done.
Done.
```

Out[102]: []

In [103…
```
%%sql
ALTER TABLE accident
ADD COLUMN accident_details_key INTEGER,
ADD CONSTRAINT fk_accident_details
    FOREIGN KEY (accident_details_key)
    REFERENCES accident_details (key);
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[103]: []

In [104…
```
%%sql
INSERT INTO accident_details (Hit_Object_in_Carriageway,Hit_Object_off_Carriageway,Jur
SELECT DISTINCT Hit_Object_in_Carriageway,Hit_Object_off_Carriageway,Junction_Locatior
FROM accident;
```

```
 * postgresql://student@/final_project_10
30623 rows affected.
```

Out[104]: []

In [105…
```
%%sql
UPDATE accident
SET accident_details_key = accident_details.key
FROM accident_details
WHERE (accident.Hit_Object_in_Carriageway = accident_details.Hit_Object_in_Carriageway
);
```

```
 * postgresql://student@/final_project_10
692698 rows affected.
```

Out[105]: []

In [106…
```
%%sql
SELECT * FROM accident
LIMIT 5;
```

```
 * postgresql://student@/final_project_10
5 rows affected.
```

Out[106]:

accident_index	age_band_of_driver	age_of_vehicle	driver_home_area_type	hit_object_in_carriageway	h
201401JI40124	46 - 55	2.0	Urban area	Previous accident	
201401JI40124	21 - 25	10.0	Data missing or out of range	None	
201401JI40124	36 - 45	7.0	Urban area	None	
201401JI40162	26 - 35	11.0	Urban area	None	

In [107…

```
%%sql
ALTER TABLE accident
DROP COLUMN age_band_of_driver,
DROP COLUMN age_of_vehicle,
DROP COLUMN driver_home_area_type,
DROP COLUMN hit_object_in_carriageway,
DROP COLUMN hit_object_off_carriageway,
DROP COLUMN junction_location,
DROP COLUMN make,
DROP COLUMN model,
DROP COLUMN sex_of_driver,
DROP COLUMN skidding_and_overturning,
DROP COLUMN vehicle_leaving_carriageway,
DROP COLUMN vehicle_manoeuvre,
DROP COLUMN vehicle_type,
DROP COLUMN vehicle_reference,
DROP COLUMN x1st_point_of_impact;
```

```
 * postgresql://student@/final_project_10
Done.
```

Out[107]: []

In [108…

```
%%sql
SELECT * FROM accident
LIMIT 5;
```

```
 * postgresql://student@/final_project_10
5 rows affected.
```

Out[108]:

accident_index	accident_type_key	conditions_key	location_key	time_key	driver_key	vehicle_details_k
201401JI40124	293	714	259524	155579	79	139
201401JI40124	293	714	259524	155579	37	879
201401JI40124	293	714	259524	155579	65	981

Ask 3: Analysis

(1) Cities with maximum number of accidents according to severity

Birmingham:

-Can be attributed to large population (more than 1 million residents)

-The city has 2.26 accidents per 1,000 people which is 32% higher than the West Midlands average of 1.7

Solution: Implement autonomous emergency braking (AEB) system in all new vehicles that automatically stops the car when it gets too close to nearby vehicles

https://www.birminghamworld.uk/news/how-many-road-accidents-are-there-in-birmingham-3668014

Wiltshire:

-Most fatal accidents due to drivers craning necks to see Stonehenge while driving on the A303 road

Solution: New plan drafted in November 2020 to move the A303 into a tunnel under the Stonehenge, this will not only eliminate the distraction for drivers, but people visiting will also be able to enjoy the historic landmark without the background noise of the road

https://www.itv.com/news/westcountry/2020-02-18/wiltshire-s-stonehenge-one-of-the-country-s-worst-landmarks-for-crashes

-For this analysis we used the PARTITION BY clause to divide the data according to 'Accident_Severity' and then using the ORDER BY clause we derived the most accidents in each category

In [109…

```
%%sql
SELECT LOCAL_AUTHORITY_DISTRICT, ACCIDENT_SEVERITY, CNT AS TOTAL_ACCIDENTS FROM
(SELECT B.LOCAL_AUTHORITY_DISTRICT, C.ACCIDENT_SEVERITY, COUNT(DISTINCT A.ACCIDENT_IND
ROW_NUMBER () OVER (PARTITION BY C.ACCIDENT_SEVERITY ORDER BY COUNT(DISTINCT A.ACCIDEN
FROM ACCIDENT A
JOIN LOCATION B
ON A.LOCATION_KEY = B.KEY
```

```
JOIN ACCIDENT_TYPE C
ON A.ACCIDENT_TYPE_KEY = C.KEY
GROUP BY B.LOCAL_AUTHORITY_DISTRICT, C.ACCIDENT_SEVERITY) A
WHERE SEQ = 1
ORDER BY TOTAL_ACCIDENTS DESC
```

```
 * postgresql://student@/final_project_10
3 rows affected.
```

Out[109]:

local_authority_district	accident_severity	total_accidents
Birmingham	Slight	6785
Birmingham	Serious	1091
Wiltshire	Fatal	69

-Birmingham has the most accidents overall, as well as described by the bar chart below:

In [110…

```
%%sql
SELECT B.LOCAL_AUTHORITY_DISTRICT,  COUNT(DISTINCT A.ACCIDENT_INDEX) TOTAL_ACCIDENTS
FROM ACCIDENT A
JOIN LOCATION B
ON A.LOCATION_KEY = B.KEY
JOIN ACCIDENT_TYPE C
ON A.ACCIDENT_TYPE_KEY = C.KEY
GROUP BY B.LOCAL_AUTHORITY_DISTRICT
ORDER BY TOTAL_ACCIDENTS DESC
LIMIT 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[110]:

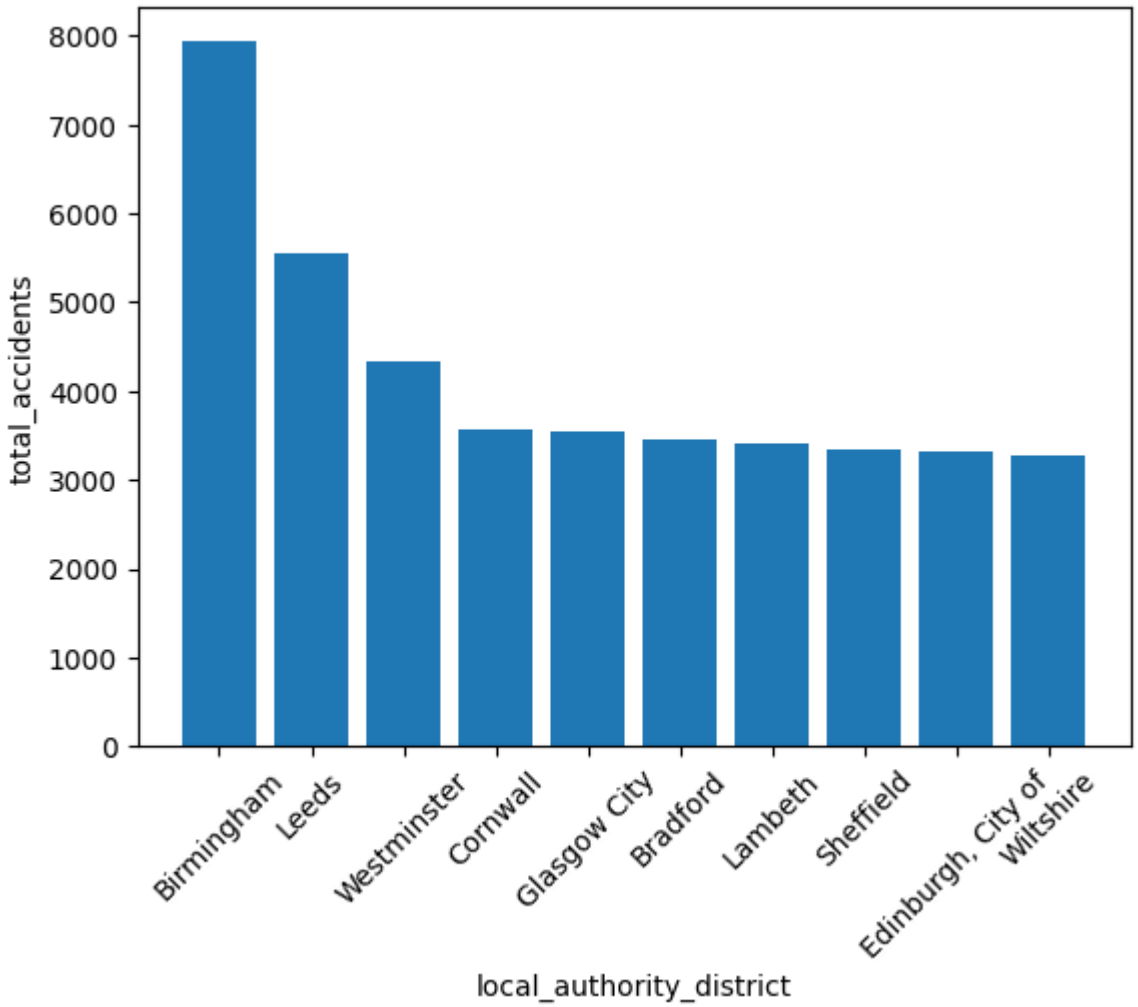
local_authority_district	total_accidents
Birmingham	7935
Leeds	5549
Westminster	4333
Cornwall	3572
Glasgow City	3535
Bradford	3449
Lambeth	3417
Sheffield	3333
Edinburgh, City of	3320
Wiltshire	3266

In [111…

```
_.bar()
```

Out[111]:

```
<BarContainer object of 10 artists>
```



(2) Most accident prone car type

Ford Fiesta:

-Fault identified in the vehicle's steering wheel that makes driver lose control of the car

-The brakes tend to become less effective after a while

Mercedes Sprinter 313 CDI:

-A software problem with the auto parking feature led to a rollaway even without an operator behind the wheel

https://www.petrolprices.com/news/john-driving-fiesta-uks-dangerous-drivers/

Solution: Recall all such vehicles to reduce the possibility of future accidents due to these faults

-We concatenated the two columns 'Make' and 'Model' in the query to derive the full name of the car to make it easier for analysis purposes

In [112…

```
%%sql
SELECT CONCAT(MAKE,' ', MODEL) AS CAR_NAME, CNT AS TOTAL_ACCIDENTS
FROM
(SELECT B.MAKE, B.MODEL, COUNT(DISTINCT ACCIDENT_INDEX) CNT
```

```
FROM ACCIDENT A
JOIN VEHICLE_DETAILS B
ON A.VEHICLE_DETAILS_KEY = B.KEY
GROUP BY MAKE, MODEL
ORDER BY COUNT(ACCIDENT_INDEX) DESC
LIMIT 10) A
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[112]:

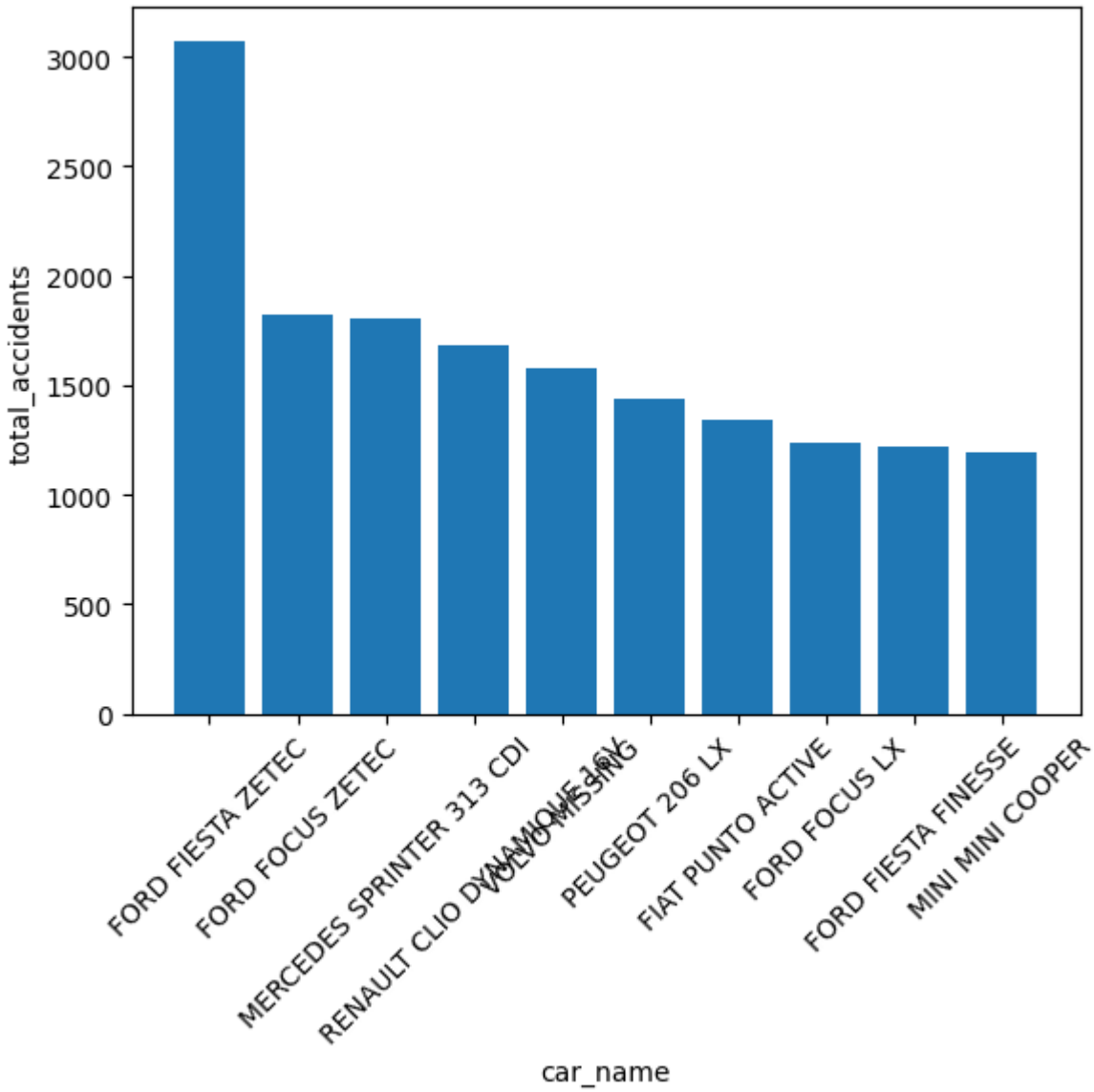
car_name	total_accidents
FORD FIESTA ZETEC	3074
FORD FOCUS ZETEC	1823
MERCEDES SPRINTER 313 CDI	1805
RENAULT CLIO DYNAMIQUE 16V	1678
VOLVO MISSING	1581
PEUGEOT 206 LX	1439
FIAT PUNTO ACTIVE	1343
FORD FOCUS LX	1241
FORD FIESTA FINESSE	1217
MINI MINI COOPER	1189

In [113…

```
_.bar()
```

Out[113]:

```
<BarContainer object of 10 artists>
```



(3) Day/time and age bracket of drivers that saw most accidents

Friday evenings appear to be the most dangerous since everyone is rushing back home after the working week, and this is further confirmed by looking at the age bracket which shows that it is mostly people aged 26 to 35 that get into these accidents

-For this analysis, we used the 'Time' column and divided the 24 hr clock into buckets using the CASE statement

https://l-a-m.org/blogs/blogs/the-uk-s-most-dangerous-drivers-revealed

In [114…

```
%%sql
SELECT B.DAY_OF_WEEK_STR,
CASE WHEN B.TIME > '00:00' AND B.TIME <= '06:00' THEN 'LATE NIGHT'
WHEN B.TIME > '06:00' AND B.TIME <= '12:00' THEN 'MORNING'
WHEN B.TIME > '12:00' AND B.TIME <= '16:00' THEN 'AFTERNOON'
WHEN B.TIME > '16:00' AND B.TIME <= '20:00' THEN 'EVENING'
WHEN B.TIME > '20:00' AND B.TIME <= '24:00' THEN 'NIGHT' ELSE NULL END AS TIME_OF_DAY,
C.LOCAL_AUTHORITY_DISTRICT, D.AGE_BAND_OF_DRIVER,
COUNT(DISTINCT A.ACCIDENT_INDEX) AS TOTAL_ACCIDENTS
```

```
FROM ACCIDENT A
JOIN DAY_TIME B
ON A.TIME_KEY = B.KEY
JOIN LOCATION C
ON A.LOCATION_KEY = C.KEY
JOIN DRIVER D
ON A.DRIVER_KEY = D.KEY
JOIN ACCIDENT_DETAILS E
ON A.ACCIDENT_DETAILS_KEY = E.KEY
GROUP BY B.DAY_OF_WEEK_STR, TIME_OF_DAY, C.LOCAL_AUTHORITY_DISTRICT, D.AGE_BAND_OF_DRI
ORDER BY TOTAL_ACCIDENTS DESC
LIMIT 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[114]:

day_of_week_str	time_of_day	local_authority_district	age_band_of_driver	total_accidents
Friday	EVENING	Birmingham	26 - 35	149
Thursday	EVENING	Birmingham	26 - 35	143
Friday	EVENING	Birmingham	Data missing or out of range	140
Tuesday	EVENING	Birmingham	26 - 35	138
Tuesday	EVENING	Birmingham	Data missing or out of range	133
Thursday	EVENING	Birmingham	Data missing or out of range	133
Monday	EVENING	Birmingham	26 - 35	130
Wednesday	EVENING	Birmingham	26 - 35	128
Wednesday	MORNING	Birmingham	26 - 35	127
Wednesday	EVENING	Birmingham	Data missing or out of range	125

In [115…

```
%%sql
SELECT B.DAY_OF_WEEK_STR,
CASE WHEN B.TIME > '00:00' AND B.TIME <= '06:00' THEN 'LATE NIGHT'
WHEN B.TIME > '06:00' AND B.TIME <= '12:00' THEN 'MORNING'
WHEN B.TIME > '12:00' AND B.TIME <= '16:00' THEN 'AFTERNOON'
WHEN B.TIME > '16:00' AND B.TIME <= '20:00' THEN 'EVENING'
WHEN B.TIME > '20:00' AND B.TIME <= '24:00' THEN 'NIGHT' ELSE NULL END AS TIME_OF_DAY,
COUNT(DISTINCT A.ACCIDENT_INDEX) AS TOTAL_ACCIDENTS
FROM ACCIDENT A
JOIN DAY_TIME B
ON A.TIME_KEY = B.KEY
JOIN LOCATION C
ON A.LOCATION_KEY = C.KEY
JOIN DRIVER D
ON A.DRIVER_KEY = D.KEY
JOIN ACCIDENT_DETAILS E
ON A.ACCIDENT_DETAILS_KEY = E.KEY
GROUP BY B.DAY_OF_WEEK_STR, TIME_OF_DAY
ORDER BY TOTAL_ACCIDENTS DESC
LIMIT 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

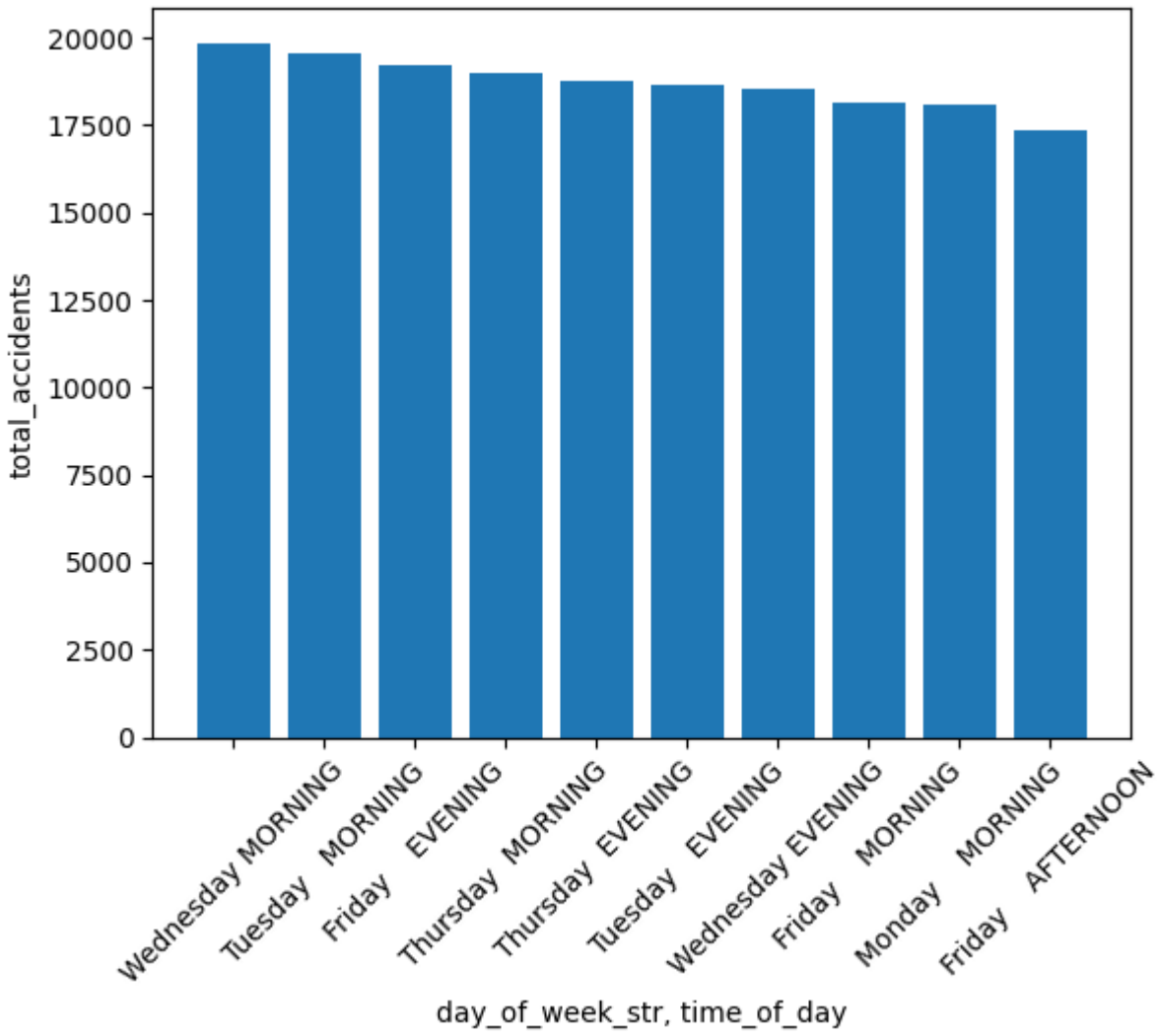
Out[115]:

day_of_week_str	time_of_day	total_accidents
Wednesday	MORNING	19862
Tuesday	MORNING	19566
Friday	EVENING	19235
Thursday	MORNING	18979
Thursday	EVENING	18740
Tuesday	EVENING	18677
Wednesday	EVENING	18518
Friday	MORNING	18154
Monday	MORNING	18096
Friday	AFTERNOON	17338

In [116…

```
_.bar()
```

Out[116]: <BarContainer object of 10 artists>



Other interesting findings in the dataset

Conditions in which top 10 highest casualties occurred

One would expect to see that the most number of casualties would be in harsh conditions but the results show that the opposite is true. In this case, we can see that majority of these accidents occurred in daylight, dry and non-windy weather.

In [117…

```
%%sql
SELECT B.LOCAL_AUTHORITY_DISTRICT, C.ACCIDENT_SEVERITY, C.NUMBER_OF_VEHICLES,
D.WEATHER_CONDITIONS, D.LIGHT_CONDITIONS, D.CARRIAGEWAY_HAZARDS,
D.ROAD_SURFACE_CONDITIONS, D.SPEED_LIMIT, MAX(C.NUMBER_OF_CASUALTIES) CASUALTIES
FROM ACCIDENT A
JOIN LOCATION B
ON A.LOCATION_KEY = B.KEY
JOIN ACCIDENT_TYPE C
ON A.ACCIDENT_TYPE_KEY = C.KEY
JOIN CONDITIONS D
ON A.CONDITIONS_KEY = D.KEY
GROUP BY  B.LOCAL_AUTHORITY_DISTRICT, C.ACCIDENT_SEVERITY, C.NUMBER_OF_VEHICLES, D.WEA
D.LIGHT_CONDITIONS, D.CARRIAGEWAY_HAZARDS, D.ROAD_SURFACE_CONDITIONS, D.SPEED_LIMIT
ORDER BY CASUALTIES DESC
LIMIT 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[117]:

local_authority_district	accident_severity	number_of_vehicles	weather_conditions	light_conditions	car
Hertsmere	Serious	2	Fine no high winds	Daylight	
County Durham	Serious	2	Fine no high winds	Daylight	
East Northamptonshire	Serious	2	Fine no high winds	Daylight	
Stroud	Serious	1	Fine no high winds	Daylight	
Herefordshire, County of	Serious	2	Fine no high winds	Daylight	
Colchester	Serious	2	Fine no high winds	Daylight	
Cherwell	Fatal	37	Fog or mist	Daylight	
Argyll and Bute	Serious	1	Fine + high winds	Daylight	
The Vale of Glamorgan	Serious	5	Fine no high winds	Daylight	
Brighton and Hove	Serious	3	Fine no high winds	Daylight	

Highest average age of vehicles by location

In [118…

```
%%sql
SELECT LOCAL_AUTHORITY_DISTRICT, AVG(AGE_OF_VEHICLE) AGE, COUNT(DISTINCT ACCIDENT_INDE
```

```
FROM(
SELECT *
FROM ACCIDENT A
JOIN LOCATION B
ON A.LOCATION_KEY = B.KEY
LEFT JOIN ACCIDENT_TYPE C
ON A.ACCIDENT_TYPE_KEY = C.KEY
LEFT JOIN VEHICLE_DETAILS D
ON A.VEHICLE_DETAILS_KEY = D.KEY
JOIN ACCIDENT_DETAILS E
ON A.ACCIDENT_DETAILS_KEY = E.KEY) A
WHERE AGE_OF_VEHICLE IS NOT NULL
GROUP BY LOCAL_AUTHORITY_DISTRICT
ORDER BY AGE DESC
LIMIT 10
```

```
 * postgresql://student@/final_project_10
10 rows affected.
```

Out[118]:

local_authority_district	age	total_accidents
Isle of Wight	9.8879159369527145	775
Thanet	9.4617330803289058	1042
Forest of Dean	9.3732970027247956	244
North Devon	9.3386308068459658	555
Weymouth and Portland	9.2963752665245203	292
Torbay	9.2370160528800755	701
Cornwall	9.2311261071277942	3134
West Devon	9.2298387096774194	345
South Hams	9.2253012048192771	557
Mid Devon	9.2200000000000000	343

Monthly analysis of total accidents

In [119…

```
%%sql
SELECT MONTH_OF_YEAR_STR, B.YEAR, MONTH_OF_YEAR, COUNT(DISTINCT A.ACCIDENT_INDEX) TOTA
FROM ACCIDENT A
JOIN DAY_TIME B
ON A.TIME_KEY = B.KEY
GROUP BY B.YEAR, MONTH_OF_YEAR, MONTH_OF_YEAR_STR
LIMIT 60
```

```
 * postgresql://student@/final_project_10
36 rows affected.
```

Out[119]:

month_of_year_str	year	month_of_year	total_accidents
January	2014	1	9698
February	2014	2	8933
March	2014	3	9833
April	2014	4	9156
May	2014	5	10164
June	2014	6	10284
July	2014	7	10686
August	2014	8	10025
September	2014	9	9704
October	2014	10	11146
November	2014	11	10983
December	2014	12	10034
January	2015	1	11601
February	2015	2	10284
March	2015	3	10939
April	2015	4	10874
May	2015	5	11441
June	2015	6	12058
July	2015	7	12771
August	2015	8	11470
September	2015	9	12201
October	2015	10	12409
November	2015	11	12378
December	2015	12	11630
January	2016	1	11688
February	2016	2	10657
March	2016	3	10836
April	2016	4	10592
May	2016	5	11482
June	2016	6	11046
July	2016	7	11777
August	2016	8	11461
September	2016	9	11571

month_of_year_str	year	month_of_year	total_accidents
October	2016	10	11624
November	2016	11	12741
December	2016	12	11146

In [120…
```
_.bar()
```

Out[120]: <BarContainer object of 36 artists>