BIG DATA ANALYTICS

ANALYSIS AND VISUALISATION OF OPEN-SOURCE POLICE DATA

ABSTRACT

A brief analysis of 'Open Source Police Data' provided on the data.gov.uk website. This site provides crime records from various counties in the United Kingdom across various time periods. Two police data sets from two areas namely Leicestershire Street and Northumbria Street are taken. Detailed analysis of crimes that occurred during the month of March 2021 is done using Apache Spark SQL, initiated by data cleansing, configurations and pre-processing. The resulting outputs and insights are pictorially showcased using graphs and charts viz bar charts, pie charts and scatter plots using snippets. This paper aims to provide a clear picture of the intensity of crimes in different cities and how they affect the safety of people's lives via big data analytics.

Table of Contents

LIST OF FIGURES

INTRODUCTION

Crime is a truly massive security concern that impacts mankind all over the world. There are numerous reasons that aid in the occurrence of crimes. Including inadequate education, wealth disparity, persecution, grievances etc. can be the causes of crimes. Crimes against individuals and possessions have become a severe hazard to people's lives and socioeconomic activities. Criminal activities can take various forms namely robbery, theft, drugs, violence and sexual offences, anti-social behaviour, burglary etc. This context of criminal domination badly affects the prosperity of society since there is a deep connection between society and development. It can also cause a threat to people's lives. It can even cause physical and mental trauma to humanity. The crime rate can vary from place to place. However crime can take place anywhere, and certain regions are prone to it. Taking these reasons into account people would have the tendency to move to safer places.

Hence in this report studies whether certain factors like geographic features, locations, time period affect the frequency of crime rate and how diverse are the crimes by comparing two localities. Also, the last outcome of the crime investigation which implies how they have been resolved is also considered for comparison. The report covers an investigation of the twelve attributes. The unnecessary variables are omitted during pre-processing and descriptive analysis is carried out to arrive at meaningful conclusions. The data sets used here are '2021-03-leicestershire-street.csv' and '2021-03-northumbria-street.csv' obtained from https://data.police.uk/data/. The entire analysis is implemented in Jupyter Notebook by using Pyspark library and SQL library. This section outlines critical insights gathered from the research that can be efficiently used for future reference.

Name	Description
Crime ID	Id of Crime
Month	Date of crime in the format yyyy-mm
Reported by	The force that provided the data
Falls within	Same as 'Reported by'
Longitude	Longitude coordinate of the crime
Latitude	Latitude coordinate of the crime
Location	Specific or near location of the crime.
LSOA code	Code of the Lower Layer Super Output Area (LSOA) where the crime was committed.
LSOA name	Name of the LSOA where the crime was committed.

Crime type	16 types of crime according to Data
Last outcome category	A reference to whichever of the outcomes associated with the crime occurred most recently.
Context	Additional data

METHODOLOGY

SETTING UP OF ENVIRONMENT

Set the necessary environment(Python is a pre-requirement) by installing Jupyter notebook in order to begin with the analysis.

Select web Console after logging into 'https://cloudxlab.com/my-lab'. The following login credentials are used to access the Web Console:

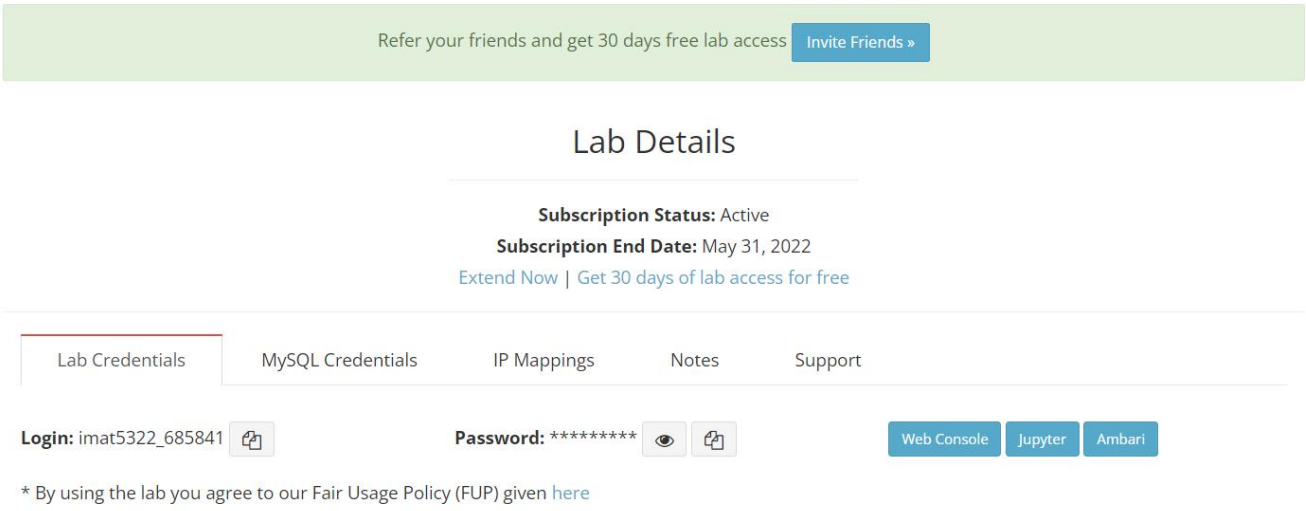


Figure 1.1 Cloudxlab

```
cxln4 login: imat5322_685841
Password:
Last login: Thu May 12 23:07:24 on pts/2
[imat5322_685841@cxln4 ~]$ hadoop fs -put '2021-03-leicestershire-street.csv'
put: `2021-03-leicestershire-street.csv': File exists
[imat5322_685841@cxln4 ~]$ hadoop fs -put '2021-03-northumbria-street.csv'
put: `2021-03-northumbria-street.csv': File exists
[imat5322_685841@cxln4 ~]$
```

Figure 1.2 Web Console

Since two data sets are considered in this report for analysis ,the data sets '2021-03-leicestershire-street.csv' and '2021-03-northumbria-street.csv' is uploaded to the Hadoop distributed file system (HDFS). A connection to the Kernel is created as a result of this operation.

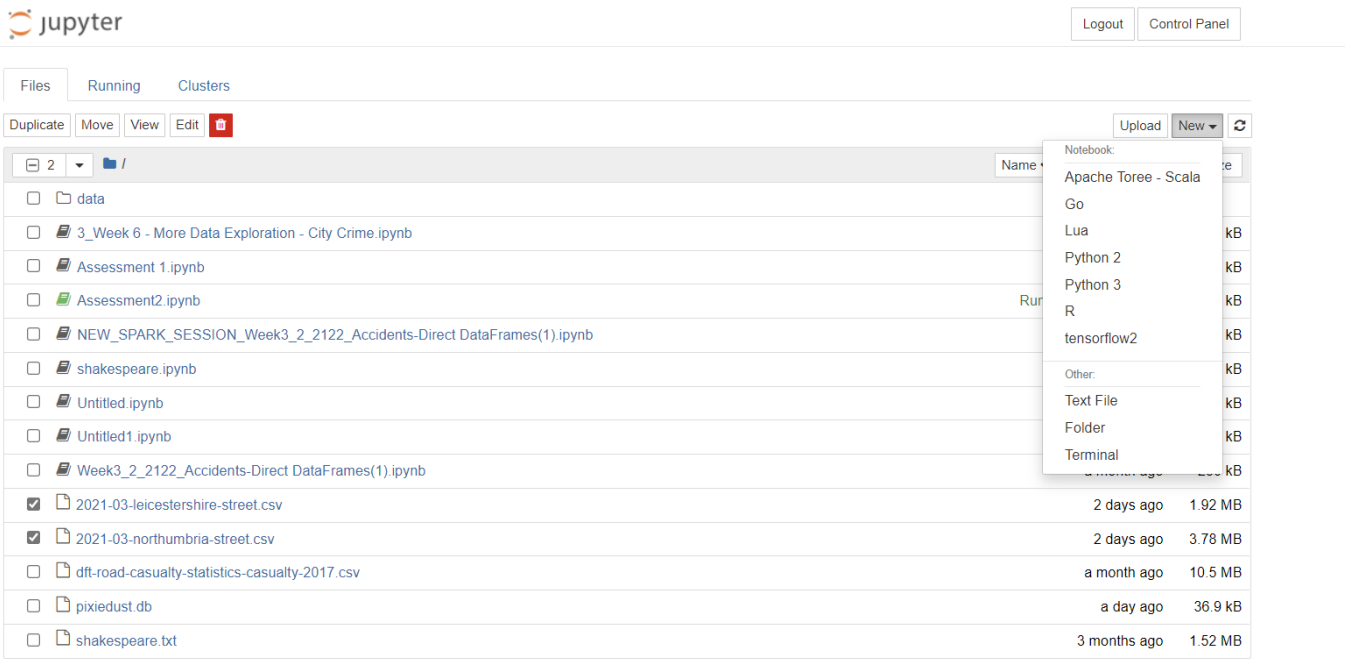


Figure 1.3 Jupyterhub

As shown in the Figure 1.3 open Jupyterhub and upload the respective data sets. Then click on 'New' and select Python3.

In [1]:
```
import os
import sys

os.environ["SPARK_HOME"] = "/usr/spark2.4.3"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"
os.environ["PYSPARK_PYTHON"] = "/usr/local/anaconda/bin/python"
os.environ["PYSPARK_DRIVER_PYTHON"] = "/usr/local/anaconda/bin/python"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

from pyspark import SparkContext, SparkConf
from pyspark.sql import SQLContext

from pyspark.sql import SparkSession
from functools import reduce
from pyspark.sql.functions import col, lit, when
#from graphframes import
```

In [2]:

```
import pyspark

from pyspark import SparkConf

from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

sqlContext = SQLContext(spark)
```

In [3]:

```
#import libraries and function
from io import StringIO
from collections import namedtuple
from pyspark.sql import Row
from pyspark.sql.types import *
from pyspark.sql.functions import *

import csv
import matplotlib.pyplot as plt
import gmplot
import pandas as pd
```

Figure 1.4 Configurations

Next step is a crucial prerequisite for performing the analysis. Apply the necessary configurations in the Jupyter notebook as given in the Figure 1.4 . This is done in order to make the commands work. Here Python act as an intermediate between user and Jupyter notebook. Import and installation of package of libraries including Pyspark, matplotlib, pandas is achieved.

DATA CLEANING AND TRANSFORMATION

Data cleaning and pre- processing is the vital in big data analysis. The procedure of rectifying or removing incorrect, inaccurate, erroneously formatted, repetitive, or missing data from a dataset is defined as data cleaning. Since the methods differ from dataset to dataset, there is no definite way to describe exact phases in process of data cleaning. On the other hand, the process of changing data from one structure or format to another referred to as data transformation.

2.1) FORMATION OF DATA FRAMES

Import and load the two .csv format data sets and other variables in order to create new data frames directly from .csv files as in the Figure 2.1.1. Headers, delimiters, and schemas were all used in the code. Also here the respective data frames of Leicestershire and Northumbria streets are displayed which provide the datatypes of the variables.

```
In [4]: #creates DataFrame directly Leicestershire police data csv file
        df_leicestershire= spark.read.option("header", "true").option("delimiter", ",")\
        .option("inferSchema", "true").csv("hdfs:///user/imat5322_685841/2021-03-leicestershire-street.csv")

In [5]: #to display DataFrame
        df_leicestershire

Out[5]: DataFrame[Crime ID: string, Month: string, Reported by: string, Falls within: string, Longitude: double, Latitude: double, Loca
        tion: string, LSOA code: string, LSOA name: string, Crime type: string, Last outcome category: string, Context: string]


In [9]: #creates DataFrame directly northumbria police data csv file
        df_northumbria1= spark.read.option("header", "true").option("delimiter", ",")\
        .option("inferSchema", "true").csv("hdfs:///user/imat5322_685841/2021-03-northumbria-street.csv")

In [10]: #to display DataFrame
         df_northumbria1

Out[10]: DataFrame[Crime ID: string, Month: string, Reported by: string, Falls within: string, Longitude: double, Latitude: double, Loca
         tion: string, LSOA code: string, LSOA name: string, Crime type: string, Last outcome category: string, Context: string]
```

Figure 2.1.1 Creation of two data frames

Now print the schema of these two data frames. This comes up with the structure(double,string,etc..) of data frames. Generally, printSchema give the explicit view of structure of the data frames.

```
In [6]: #to print schema of the DataFrame
        df_leicestershire.printSchema()

root
 |-- Crime ID: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- Reported by: string (nullable = true)
 |-- Falls within: string (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Location: string (nullable = true)
 |-- LSOA code: string (nullable = true)
 |-- LSOA name: string (nullable = true)
 |-- Crime type: string (nullable = true)
 |-- Last outcome category: string (nullable = true)
 |-- Context: string (nullable = true)


In [11]: #to print schema of the DataFrame
         df_northumbria1.printSchema()

root
 |-- Crime ID: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- Reported by: string (nullable = true)
 |-- Falls within: string (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Location: string (nullable = true)
 |-- LSOA code: string (nullable = true)
 |-- LSOA name: string (nullable = true)
 |-- Crime type: string (nullable = true)
 |-- Last outcome category: string (nullable = true)
 |-- Context: string (nullable = true)
```

Figure 2.1.2 Printing schema of dataframe

Moreover, use the function ‘take()’ to return the required number of successive rows. Here ‘take(6)’ is used to return first six rows from the data frames by default.

```
In [7]: #to return first six rows
        df_leicestershire.take(6)
```

```
Out[7]: [Row(Crime ID=None, Month='2021-03', Reported by='Leicestershire Police', Falls within='Leicestershire Police', Longitude=-1.21
        0015, Latitude=52.62141, Location='On or near Hobill Close', LSOA code='E01025631', LSOA name='Blaby 002A', Crime type='Anti-so
        cial behaviour', Last outcome category=None, Context=None),
         Row(Crime ID='d6aa170f22d346877589643f030399e22cf990385fb7b7bfa7758510a35a9f8d', Month='2021-03', Reported by='Leicestershire
        Police', Falls within='Leicestershire Police', Longitude=-1.214176, Latitude=52.621663, Location='On or near Lowland Avenue', L
        SOA code='E01025631', LSOA name='Blaby 002A', Crime type='Burglary', Last outcome category='Investigation complete; no suspect
        identified', Context=None),
         Row(Crime ID='f35db016e317df79fe1b7410bc316ae5b80ef4dd59257cc15cb28576a314707a', Month='2021-03', Reported by='Leicestershire
        Police', Falls within='Leicestershire Police', Longitude=-1.21556, Latitude=52.619443, Location='On or near Lancelot Close', LS
        OA code='E01025631', LSOA name='Blaby 002A', Crime type='Other theft', Last outcome category='Investigation complete; no suspec
        t identified', Context=None),
         Row(Crime ID='ac17ff0d0dbdbfdc48cda0fe36fe215c2e46efcad63d4e40dabd2a9436de7f57', Month='2021-03', Reported by='Leicestershire
        Police', Falls within='Leicestershire Police', Longitude=-1.212828, Latitude=52.622715, Location='On or near Kings Drive', LSOA
        code='E01025631', LSOA name='Blaby 002A', Crime type='Other theft', Last outcome category='Investigation complete; no suspect i
        dentified', Context=None),
         Row(Crime ID='de0b9fa13a7d37a8326389750236ada6c48acb64b25897e876598d13326c0115', Month='2021-03', Reported by='Leicestershire
        Police', Falls within='Leicestershire Police', Longitude=-1.21556, Latitude=52.619443, Location='On or near Lancelot Close', LS
        OA code='E01025631', LSOA name='Blaby 002A', Crime type='Other theft', Last outcome category='Investigation complete; no suspec
        t identified', Context=None),
         Row(Crime ID='e1d9a2447154862b761b3bf522fe900a6856656032a9a8cca0e1ad6e4378572f', Month='2021-03', Reported by='Leicestershire
        Police', Falls within='Leicestershire Police', Longitude=-1.214176, Latitude=52.621663, Location='On or near Lowland Avenue', L
        SOA code='E01025631', LSOA name='Blaby 002A', Crime type='Violence and sexual offences', Last outcome category='Investigation c
        omplete; no suspect identified', Context=None)]
```

```
In [12]: #to return first six rows
         df_northumbria1.take(6)
```

```
Out[12]: [Row(Crime ID=None, Month='2021-03', Reported by='Northumbria Police', Falls within='Northumbria Police', Longitude=-2.575411,
         Latitude=54.991255, Location='On or near B6318', LSOA code='E01019225', LSOA name='Carlisle 002D', Crime type='Anti-social beha
         viour', Last outcome category=None, Context=None),
          Row(Crime ID='865f6992570a3d086ea666114c240489bed719cac635f26faf8f5b2474d9855a', Month='2021-03', Reported by='Northumbria Pol
         ice', Falls within='Northumbria Police', Longitude=-1.783738, Latitude=54.899283, Location='On or near Parklands', LSOA code='E
         01020655', LSOA name='County Durham 003D', Crime type='Criminal damage and arson', Last outcome category='Investigation complet
         e; no suspect identified', Context=None),
          Row(Crime ID='46704d6ed821f234e6186eb85a154303920fdb2430df51a2844fe4dd4d7c1007', Month='2021-03', Reported by='Northumbria Pol
         ice', Falls within='Northumbria Police', Longitude=-1.783738, Latitude=54.899283, Location='On or near Parklands', LSOA code='E
         01020655', LSOA name='County Durham 003D', Crime type='Other theft', Last outcome category='Investigation complete; no suspect
         identified', Context=None),
          Row(Crime ID='2d90958c48d48a9280ae8cad4b23b77c560144683c50025c572851548613e5ec', Month='2021-03', Reported by='Northumbria Pol
         ice', Falls within='Northumbria Police', Longitude=-1.676736, Latitude=54.903395, Location='On or near Burdon Plain', LSOA code
         ='E01020663', LSOA name='County Durham 004C', Crime type='Other theft', Last outcome category='Investigation complete; no suspe
         ct identified', Context=None),
          Row(Crime ID=None, Month='2021-03', Reported by='Northumbria Police', Falls within='Northumbria Police', Longitude=-1.560279,
         Latitude=54.875639, Location='On or near Lambton Court', LSOA code='E01020611', LSOA name='County Durham 007D', Crime type='Ant
         i-social behaviour', Last outcome category=None, Context=None),
          Row(Crime ID='419ef4b3597c12c4d51ecf794ce512e0a6085a67d9185b24f26f363b7316662a', Month='2021-03', Reported by='Northumbria Pol
         ice', Falls within='Northumbria Police', Longitude=-1.56913, Latitude=54.880092, Location='On or near Vigo Lane', LSOA code='E0
         1020611', LSOA name='County Durham 007D', Crime type='Drugs', Last outcome category='Investigation complete; no suspect identif
         ied', Context=None)]
```

Figure 2.1.3 take() for two dataframes

Furthermore, for getting the count of the dataset use the function ‘count()’. And here the count of the data sets Leicestershire-street and Northumbria-street is obtained as 8113 and 18151 respectively.

```
In [8]: #to get count of the dataset
        df_leicestershire.count()
```

```
Out[8]: 8113
```

```
In [13]: #to get count of the dataset
         df_northumbria1.count()
```

```
Out[13]: 18151
```

Figure 2.1.4 Count of two data frames

2.2) NULL / MISSING VALUES OF DATA SETS

Many intriguing datasets, particularly, contain some missing data which are notified in various ways. When no information is provided for one or more elements, or for the entire unit, this is referred to as missing data. Commonly the missing data is referred as null, NaN OR NA values. By either filling the null space or deleting the entire row including the missing value, missing data can be erased. It raises the level of ambiguity of analysis and its storage consuming. The following queries find the count of the missing values in the two data sets as a part of data cleansing and 'while' loop statement used for this.

```
In [14]: #checking count of null values
         i = 0 # reset counter
         #get the total records amount
         total = df_leicestershire.count()

         print("Total Records = " + str(total))

         #print the amount of columns
         print("Total columns = " + str(len(df_leicestershire.columns)))
         print("-----------------------------------------")

         #loop entire table and get the missing value number and missing rate of each column
         while i < len(df_leicestershire.columns): #loop through all columns
             print(str(i+1) + "." + str(df_leicestershire[i]))
             print("  Missing Values = ")
             print("-----------------------------------------")
             mv = df_leicestershire.select([count(when(df_leicestershire[i].isNull(),\
                                                 True))]).show() #check for missing values
             i = i+1 #counter add 1
```

```
Total Records = 8113
Total columns = 12
-----------------------------------------
1.Column<b'Crime ID'>
  Missing Values =
-----------------------------------------
+-------------------------------------------------+
|count(CASE WHEN (Crime ID IS NULL) THEN true END)|
+-------------------------------------------------+
|                                              888|
+-------------------------------------------------+

2.Column<b'Month'>
  Missing Values =
-----------------------------------------
+-----------------------------------------------+
|count(CASE WHEN (Month IS NULL) THEN true END)|
+-----------------------------------------------+
|                                              0|
```

Figure 2.2.1 Count of missing values in Leicestershire dataset

In the case of Leicestershire, total records is 8113 and total columns Is 12. Here the column 'Context' has the maximum number of missing values which is 8113. This is because the whole column is blank in the data set. The second largest count of missing values is 888 which is given by 'Crime ID' and 'Last outcome category'. This implies that the required information are not recorded or missing for these attributes. The columns 'Month', 'Reported by', 'Falls within' , 'Location' and 'Crime Type' has no missing values.

```
In [15]:
i = 0 # reset counter
#get the total records amount
total = df_northumbria1.count()

print("Total Records = " + str(total))

#print the amount of columns
print("Total columns = " + str(len(df_northumbria1.columns)))
print("-----------------------------------------")

#loop entire table and get the missing value number and missing rate of each column
while i < len(df_northumbria1.columns): #loop through all columns
    print(str(i+1) + "." + str(df_northumbria1[i]))
    print("  Missing Values = ")
    print("-----------------------------------------")
    mv = df_northumbria1.select([count(when(df_northumbria1[i].isNull(),\
                                        True))]).show() #check for missing values
    i = i+1 #counter add 1
```

```
Total Records = 18151
Total columns = 12
-----------------------------------------
1.Column<b'Crime ID'>
  Missing Values =
-----------------------------------------
+-----------------------------------------------+
|count(CASE WHEN (Crime ID IS NULL) THEN true END)|
+-----------------------------------------------+
|                                           6860|
+-----------------------------------------------+

2.Column<b'Month'>
  Missing Values =
-----------------------------------------
+-----------------------------------------------+
|count(CASE WHEN (Month IS NULL) THEN true END)|
+-----------------------------------------------+
|                                              0|
```

Figure 2.2.2 count of missing values in Northumbria dataset

For the above case of Northumbria, the data set contains 18151 records and 12 columns. Here also the column ‘Context’ has the maximum number of missing values which is 18151. The columns ‘Crime ID’ and ’Last outcome category’ has the second maximum number of missing values given by 6860. The rest of the columns have no missing values.

Therefore, it can be concluded that the dataset of Leicestershire has only ~11% of missing records in the attribute ‘Last outcome category’, whereas the dataset of Northumbria has ~38%. It might have occurred either because the investigation conducted was not recorded or because it was never conducted.

2.3) RENAMING THE COLUMN NAMES

In this step, the column names are changed and displayed in a tabular form using the function ‘show()’. The top 20 rows are shown below. This is the simplest technique to rename one column at a time.. The syntax is given by, ‘new data frame=previous dataframe.withColumnRenamed(“existing column name”,“new column name”)’

In [16]:
```
# to rename the column names
leicestershire_20211=df_leicestershire.withColumnRenamed("Crime ID","Crime_ID")
leicestershire_20212=leicestershire_20211.withColumnRenamed("Month","Month_March")
leicestershire_20213=leicestershire_20212.withColumnRenamed("Reported by","Reported_by")
leicestershire_20214=leicestershire_20213.withColumnRenamed("Falls within","Falls_within")
leicestershire_20215=leicestershire_20214.withColumnRenamed("Longitude","longitude")
leicestershire_20216=leicestershire_20215.withColumnRenamed("Latitude","latitude")
leicestershire_20217=leicestershire_20216.withColumnRenamed("Location","location")
leicestershire_20218=leicestershire_20217.withColumnRenamed("LSOA code","LSOA_code")
leicestershire_20219=leicestershire_20218.withColumnRenamed("LSOA name","LSOA_name")
leicestershire_202110=leicestershire_20219.withColumnRenamed("Crime type","Crime_type")
leicestershire_202111=leicestershire_202110.withColumnRenamed("Last outcome category","Last_outcome")
leicestershire_202112=leicestershire_202111.withColumnRenamed("Context","context")
```

In [17]:
```
#to show the dataframe
leicestershire_202112.show()
```

```
+--------------------+-----------+--------------------+--------------------+---------+---------+--------------------+---------+
----------+--------------------+--------------------+-------+
|            Crime_ID|Month_March|         Reported_by|       Falls_within|longitude| latitude|            location|LSOA_code|
LSOA_name|          Crime_type|       Last_outcome|context|
+--------------------+-----------+--------------------+--------------------+---------+---------+--------------------+---------+
----------+--------------------+--------------------+-------+
|                null|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.210015| 52.62141|On or near Hobill...|E01025631|
Blaby 002A|Anti-social behav...|                null|   null|
|d6aa170f22d346877...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.214176|52.621663|On or near Lowlan...|E01025631|
Blaby 002A|            Burglary|Investigation com...|   null|
|f35db016e317df79f...|    2021-03|Leicestershire Po...|Leicestershire Po...| -1.21556|52.619443|On or near Lancel...|E01025631|
Blaby 002A|         Other theft|Investigation com...|   null|
|ac17ff0d0dbdbfdc4...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.212828|52.622715|On or near Kings ...|E01025631|
Blaby 002A|         Other theft|Investigation com...|   null|
|de0b9fa13a7d37a83...|    2021-03|Leicestershire Po...|Leicestershire Po...| -1.21556|52.619443|On or near Lancel...|E01025631|
Blaby 002A|         Other theft|Investigation com...|   null|
|e1d9a2447154862b7...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.214176|52.621663|On or near Lowlan...|E01025631|
Blaby 002A|Violence and sexu...|Investigation com...|   null|
|cbef62e2c348e7892...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.210015| 52.62141|On or near Hobill...|E01025631|
Blaby 002A|Violence and sexu...|Unable to prosecu...|   null|
|                null|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.225664|52.616057|On or near Yew Close|E01025632|
Blaby 002B|Anti-social behav...|                null|   null|
|3fd7d94146f2ddfad...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.217867|52.618424|On or near Excali...|E01025632|
Blaby 002B|Criminal damage a...|Investigation com...|   null|
|83a20fd73aedc87f6...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.216845|52.619442|On or near Merlin...|E01025632|
Blaby 002B|         Other theft|Investigation com...|   null|
|b285e24cfe3538e83...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.217867|52.618424|On or near Excali...|E01025632|
Blaby 002B|             Robbery|Awaiting court ou...|   null|
|cd3c533366473530a...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.217867|52.618424|On or near Excali...|E01025632|
Blaby 002B|Violence and sexu...|Unable to prosecu...|   null|
|                null|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.206075|52.623632|On or near South ...|E01025633|
Blaby 002C|Anti-social behav...|                null|   null|
|                null|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.206075|52.623632|On or near South ...|E01025633|
Blaby 002C|Anti-social behav...|                null|   null|
|26c513a88bf14681b...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.209332| 52.62653|On or near Packer...|E01025633|
Blaby 002C|Criminal damage a...|Investigation com...|   null|
|2bc6499a27919014f...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.206075|52.623632|On or near South ...|E01025633|
Blaby 002C|Criminal damage a...|Investigation com...|   null|
|05257916a2937bec8...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.202829|52.621749|On or near Garden...|E01025633|
Blaby 002C|               Drugs|    Local resolution|   null|
|cb7d92d33d0c3af57...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.206713|52.619402|On or near Petrol...|E01025633|
Blaby 002C|         Other theft|Investigation com...|   null|
|f7fc32b7242a3245b...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.206713|52.619402|On or near Petrol...|E01025633|
Blaby 002C|       Vehicle crime|Investigation com...|   null|
|40d692fe55a016fd1...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.206713|52.619402|On or near Petrol...|E01025633|
Blaby 002C|       Vehicle crime|Investigation com...|   null|
+--------------------+-----------+--------------------+--------------------+---------+---------+--------------------+---------+
----------+--------------------+--------------------+-------+
only showing top 20 rows
```

Figure 2.3.1 Column renamed data frame of Leicestershire

The resulting new data frame obtained has the name Leicestershire_202112

```
In [18]: #to rename the column names
northumbria1_2021=df_northumbria1.withColumnRenamed("Crime ID","Crime_ID")
northumbria1_20212=northumbria1_2021.withColumnRenamed("Month","Month_March")
northumbria1_20213=northumbria1_20212.withColumnRenamed("Reported by","Reported_by")
northumbria1_20214=northumbria1_20213.withColumnRenamed("Falls within","Falls_within")
northumbria1_20215=northumbria1_20214.withColumnRenamed("Longitude","longitude")
northumbria1_20216=northumbria1_20215.withColumnRenamed("Latitude","latitude")
northumbria1_20217=northumbria1_20216.withColumnRenamed("Location","location")
northumbria1_20218=northumbria1_20217.withColumnRenamed("LSOA code","LSOA_code")
northumbria1_20219=northumbria1_20218.withColumnRenamed("LSOA name","LSOA_name")
northumbria1_202110=northumbria1_20219.withColumnRenamed("Crime type","Crime_type")
northumbria1_202111=northumbria1_202110.withColumnRenamed("Last outcome category","Last_outcome")
northumbria1_202112=northumbria1_202111.withColumnRenamed("Context","context")

In [19]: #to show the dataframe
northumbria1_202112.show()
```

```
+--------------------+-----------+--------------------+--------------------+---------+---------+--------------------+---------+---------+
|            Crime_ID|Month_March|         Reported_by|      Falls_within|longitude| latitude|            location|LSOA_code|LSOA_name|
|          Crime_type|       Last_outcome|context|
+--------------------+-----------+--------------------+--------------------+---------+---------+--------------------+---------+---------+
|                null|    2021-03|Northumbria Police|Northumbria Police|-2.575411|54.991255|    On or near B6318|E01019225|Carlisle 002D|Anti-social behav...|                null|   null|
|865f6992570a3d086...|    2021-03|Northumbria Police|Northumbria Police|-1.783738|54.899283|On or near Parklands|E01020655|County Durham 003D|Criminal damage a...|Investigation com...|   null|
|46704d6ed821f234e...|    2021-03|Northumbria Police|Northumbria Police|-1.783738|54.899283|On or near Parklands|E01020655|County Durham 003D|         Other theft|Investigation com...|   null|
|2d90958c48d48a928...|    2021-03|Northumbria Police|Northumbria Police|-1.676736|54.903395|On or near Burdon...|E01020663|County Durham 004C|         Other theft|Investigation com...|   null|
|                null|    2021-03|Northumbria Police|Northumbria Police|-1.560279|54.875639|On or near Lambto...|E01020611|County Durham 007D|Anti-social behav...|                null|   null|
|419ef4b3597c12c4d...|    2021-03|Northumbria Police|Northumbria Police| -1.56913|54.880092|On or near Vigo Lane|E01020611|County Durham 007D|               Drugs|Investigation com...|   null|
|431a0075a88aa47cf...|    2021-03|Northumbria Police|Northumbria Police| -1.56913|54.880092|On or near Vigo Lane|E01020611|
```

Figure 2.3.2 Column renamed data frame of Northumbria

Here also the changed name of the new data frame northumbria_202112 is noticeable.

2.4) FINDING THE SUMMARY OF COLUMNS

In this step, the summary of each column of two data sets that is, count, mean, standard deviation, minimum and maximum are found. Here also while loop statement is used for finding missing values.

```
In [20]: #to find count,mean,stddev,min,max
#reset counter
i=0

print("----------------------------------------")

#loop each column and get the description of each one
while i < len(leicestershire_202112.columns):
    mv = leicestershire_202112.where(leicestershire_202112[i] =='').count() #get the missing value number of current column
    print(str(i+1) + "." + str(leicestershire_202112[i]))
    leicestershire_202112.where(leicestershire_202112[i] !='').describe(leicestershire_202112.columns[i]).show()
    print("----------------------------------------")
    i = i+1 #counter add 1
```

```
----------------------------------------
1.Column<b'Crime_ID'>
+-------+--------------------+
|summary|            Crime_ID|
+-------+--------------------+
|  count|                7225|
|   mean|                null|
| stddev|                null|
|    min|000025d45f1ff2db8...|
|    max|fffec74c7bd120b5c...|
+-------+--------------------+

----------------------------------------
2.Column<b'Month_March'>
+-------+-----------+
|summary|Month_March|
+-------+-----------+
|  count|       8113|
|   mean|       null|
```

Figure 2.4.1 Summary of Leicestershire dataset

```
In [21]: #to find count,mean,stddev,min,max
         #reset counter
         i=0

         print("----------------------------------------")

         #Loop each column and get the descriptionof each one
         while i < len(northumbria1_202112.columns):
             mv = northumbria1_202112.where(northumbria1_202112[i] =='').count() #get the missing value number of current column
             print(str(i+1) + "." + str(northumbria1_202112[i]))
             northumbria1_202112.where(northumbria1_202112[i] !='').describe(northumbria1_202112.columns[i]).show()
             print("----------------------------------------")
             i = i+1 #counter add 1
```

```
----------------------------------------
1.Column<b'Crime_ID'>
+-------+--------------------+
|summary|            Crime_ID|
+-------+--------------------+
|  count|               11291|
|   mean|                null|
| stddev|                null|
|    min|00005d26c37f64ba8...|
|    max|fffabd91048423dc7...|
+-------+--------------------+

----------------------------------------
2.Column<b'Month_March'>
+-------+-----------+
|summary|Month_March|
+-------+-----------+
|  count|      18151|
|   mean|       null|
```

Figure 2.4.2 Summary of Northumbria dataset

Here the count of columns is the count value obtained after subtracting the null values from its actual count. For Leicestershire &Northumbria data the columns ‘longitude’, ‘latitude’ and ‘context’ has 0 ‘count’ values. The mean and standard deviation is null for all the columns because they are categorical variables. And the columns ‘longitude’, ‘latitude’ and ‘context’ has no minimum and maximum values. But rest of the columns has minimum and maximum values.

2.5) REGISTERING DATAFRAME AS TABLE

In this step construct two new data frames by removing the unwanted columns/variables using SQL queries. For that, a SQL context , which is the primary access point for Spark SQL function, is created. Spark SQL is a Spark Core component that enables the analysis of structured and semi-structured data. The syntax is, sqlContext.registerDataFrameAsTable(dataframe name,”sql table name”). Execute the SQL queries given below to obtain the following results:

```
In [22]: #to register DataFrame as table
         sqlContext.registerDataFrameAsTable(leicestershire_202112,"leicestershire_2021")

In [23]: sqlContext.sql("select * from leicestershire_2021")

Out[23]: DataFrame[Crime_ID: string, Month_March: string, Reported_by: string, Falls_within: string, longitude: double, latitude: doubl
         e, location: string, LSOA_code: string, LSOA_name: string, Crime_type: string, Last_outcome: string, context: string]

In [25]: #selecting required columns & showing first five rows
         sqlContext.sql("select Reported_by, Falls_within, longitude, latitude, location,\
                         Crime_type, Last_outcome from leicestershire_2021 group\
                        by Reported_by, Falls_within, longitude, latitude, location,\
                        Crime_type, Last_outcome").show(5)
```

```
+--------------------+--------------------+---------+---------+--------------------+--------------------+--------------------+
|         Reported_by|        Falls_within|longitude| latitude|            location|          Crime_type|        Last_outcome|
+--------------------+--------------------+---------+---------+--------------------+--------------------+--------------------+
|Leicestershire Po...|Leicestershire Po...|-1.196626|52.618462|On or near Royce ...|       Vehicle crime|Investigation com...|
|Leicestershire Po...|Leicestershire Po...| -1.20127|52.773874|On or near Cradoc...|               Drugs|Status update una...|
|Leicestershire Po...|Leicestershire Po...|-1.220907|52.773249|On or near Cumber...|Violence and sexu...|Investigation com...|
|Leicestershire Po...|Leicestershire Po...|-1.210446|52.751695|On or near Acer C...|Anti-social behav...|                null|
|Leicestershire Po...|Leicestershire Po...|-1.223841|52.752475|On or near Parkin...|       Vehicle crime|Investigation com...|
+--------------------+--------------------+---------+---------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

```
In [27]: #to register DataFrame as table
         sqlContext.registerDataFrameAsTable(northumbria1_202112,"northumbria1_2021")

In [28]: sqlContext.sql("select * from northumbria1_2021")

Out[28]: DataFrame[Crime_ID: string, Month_March: string, Reported_by: string, Falls_within: string, longitude: double, latitude: doubl
         e, location: string, LSOA_code: string, LSOA_name: string, Crime_type: string, Last_outcome: string, context: string]

In [30]: #selecting required columns & showing first five rows
         sqlContext.sql("select Reported_by, Falls_within, longitude, latitude, location,\
                         Crime_type, Last_outcome from northumbria1_2021 group\
                        by Reported_by, Falls_within, longitude, latitude, location,\
                        Crime_type, Last_outcome").show(5)
```

```
+----------------+----------------+---------+---------+--------------------+--------------------+--------------------+
|     Reported_by|   Falls_within|longitude| latitude|            location|          Crime_type|        Last_outcome|
+----------------+----------------+---------+---------+--------------------+--------------------+--------------------+
|Northumbria Police|Northumbria Police|-1.543545| 54.95393|On or near Superm...|         Shoplifting|Investigation com...|
|Northumbria Police|Northumbria Police|-1.615856|54.948994|On or near Trevet...|Violence and sexu...|Status update una...|
|Northumbria Police|Northumbria Police|-1.589307|54.952817|On or near Avon S...|Violence and sexu...|Status update una...|
|Northumbria Police|Northumbria Police|-1.639851|54.945707|On or near Dougla...|Criminal damage a...|Investigation com...|
|Northumbria Police|Northumbria Police|-1.537985|54.945911| On or near Tuneside|Violence and sexu...|Status update una...|
+----------------+----------------+---------+---------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Figure 2.5.1 SQL tables of two datasets

Here only first 5 rows are returned for the two datasets.

2.6) CALCULATING COUNT OF VALUES OF COLUMNS

Furthermore, the count of the Crime_type is found using SQL context.

```
In [36]: #to find count of values of a specific column
         sqlContext.sql("select Crime_type, count(*) as No_of_Crime_type from\
                         leicestershire_2021 group by Crime_type order by Crime_type").show()
```

```
+--------------------+----------------+
|          Crime_type|No_of_Crime_type|
+--------------------+----------------+
|Anti-social behav...|             888|
|       Bicycle theft|             110|
|            Burglary|             294|
|Criminal damage a...|             749|
|               Drugs|             266|
|         Other crime|             211|
|         Other theft|             552|
|Possession of wea...|              83|
|        Public order|             894|
|             Robbery|              50|
|         Shoplifting|             242|
|Theft from the pe...|              37|
|       Vehicle crime|             458|
|Violence and sexu...|            3279|
+--------------------+----------------+
```

```
In [39]: sqlContext.sql("select Crime_type, count(*) as No_of_Crime_type from\
                         northumbria1_2021 group by Crime_type order by Crime_type").show()
```

```
+--------------------+----------------+
|          Crime_type|No_of_Crime_type|
+--------------------+----------------+
|Anti-social behav...|            6860|
|       Bicycle theft|              95|
|            Burglary|             468|
|Criminal damage a...|            1664|
|               Drugs|             355|
|         Other crime|             390|
|         Other theft|             839|
|Possession of wea...|              96|
|        Public order|            1475|
|             Robbery|              51|
|         Shoplifting|             650|
|Theft from the pe...|              66|
|       Vehicle crime|             460|
|Violence and sexu...|            4682|
+--------------------+----------------+
```

Figure 2.6.1 Count of each values of Crime type of two datasets

By analysing Figure 2.6.1, in the case of Leicestershire, ‘violence and sexual offences’ have the highest count i.e. 3279 And ‘theft from the person’ has the lowest count which is 37.

And for Northumbria, it is understood that ‘Anti-social behaviour’ has the highest count that is, 6860. And the lowest count is 51 which is the Crime type ‘Robbery’.

This implies that spatial variation can cause variation in crime rates.

2.7) ARRANGING THE COUNT IN ASCENDING AND DESCENDING ORDER

As earlier, in this step, besides the calculation of the count of each values, those counts are arranged in ascending order as shown in the Figure 2.7.1. And this is tested for the two SQL tables.

```
In [42]: #to count columns in ascending order
         sqlContext.sql("select Crime_type, \
                         count(*)\
                         from leicestershire_2021 \
                         group by Crime_type\
                         order by count(*) asc").show()
```

```
+--------------------+--------+
|          Crime_type|count(1)|
+--------------------+--------+
|Theft from the pe...|      37|
|             Robbery|      50|
|Possession of wea...|      83|
|       Bicycle theft|     110|
|         Other crime|     211|
|         Shoplifting|     242|
|               Drugs|     266|
|            Burglary|     294|
|       Vehicle crime|     458|
|         Other theft|     552|
|Criminal damage a...|     749|
|Anti-social behav...|     888|
|        Public order|     894|
|Violence and sexu...|    3279|
+--------------------+--------+
```

```
In [44]: sqlContext.sql("select Crime_type, \
                         count(*)\
                         from northumbria1_2021 \
                         group by Crime_type\
                         order by count(*) asc").show()
```

```
+--------------------+--------+
|          Crime_type|count(1)|
+--------------------+--------+
|             Robbery|      51|
|Theft from the pe...|      66|
|       Bicycle theft|      95|
|Possession of wea...|      96|
|               Drugs|     355|
|         Other crime|     390|
|       Vehicle crime|     460|
|            Burglary|     468|
|         Shoplifting|     650|
|         Other theft|     839|
|        Public order|    1475|
|Criminal damage a...|    1664|
|Violence and sexu...|    4682|
|Anti-social behav...|    6860|
+--------------------+--------+
```

Figure 2.7.1 Ascending order of count

Similarly the count obtained can be arranged in descending order also as follows:

Here the count of the column ‘Last outcome category’ of two places are considered.

```
In [43]: #to count columns in descending order
         sqlContext.sql("select Last_outcome, \
                        count(*)\
                        from leicestershire_2021 \
                        group by Last_outcome\
                        order by count(*) desc").show()
```

```
+--------------------+--------+
|        Last_outcome|count(1)|
+--------------------+--------+
|Unable to prosecu...|    3089|
|Investigation com...|    2672|
|                null|     888|
|    Local resolution|     456|
|Court result unav...|     331|
|Status update una...|     159|
|Action to be take...|     125|
|Further action is...|      92|
|Awaiting court ou...|      89|
|Formal action is ...|      89|
|Further investiga...|      78|
|Offender given a ...|      45|
+--------------------+--------+
```

```
In [45]: sqlContext.sql("select Last_outcome, \
                        count(*)\
                        from northumbria1_2021 \
                        group by Last_outcome\
                        order by count(*) desc").show()
```

```
+--------------------+--------+
|        Last_outcome|count(1)|
+--------------------+--------+
|                null|    6860|
|Investigation com...|    5664|
|Status update una...|    4807|
|Court result unav...|     545|
|Offender given a ...|     170|
|Offender given a ...|      38|
|    Local resolution|      27|
|Awaiting court ou...|      22|
|Offender given pe...|      11|
|Suspect charged a...|       7|
+--------------------+--------+
```

Figure 2.7.2 Descending order of count

This figure implies that, for Leicestershire, the last outcome category- Unable to prosecute suspect has the largest count which is 3089. i.e., in most of the cases, the suspect is not prosecuted. And there are only the least number of cases where the offender given a caution which is counted as 45. For Northumbria in the highest number of cases, the last outcome category is null. I.e., no information is provided. And there are only 7 cases where the Suspect charged as part of another case. Therefore, this arrangement helps to interpret the largest and smallest count.

Furthermore, another SQL query to provide a unique output is executed. The syntax is given by, '"select * from the table where variable== 'condition'".

```
In [46]: # displaying crime type together with last outcome for specific location
         sqlContext.sql("select Crime_type, Last_outcome from leicestershire_2021 where location=='On or near A607'").show()
```

```
+-------------+--------------------+
|   Crime_type|        Last_outcome|
+-------------+--------------------+
|     Burglary|Investigation com...|
|  Other theft|Investigation com...|
|Vehicle crime|Investigation com...|
|Vehicle crime|Investigation com...|
+-------------+--------------------+
```

```
In [85]: sqlContext.sql("select Crime_type, Last_outcome from northumbria1_2021 where location=='On or near A19'").show()

+--------------------+--------------------+
|          Crime_type|        Last_outcome|
+--------------------+--------------------+
|Anti-social behav...|                null|
|Anti-social behav...|                null|
|        Public order|Status update una...|
|Violence and sexu...|Status update una...|
|Anti-social behav...|                null|
|        Public order|Investigation com...|
|Anti-social behav...|                null|
|Anti-social behav...|                null|
|        Public order|Status update una...|
|Violence and sexu...|Court result unav...|
|Anti-social behav...|                null|
|         Other crime|Status update una...|
|Violence and sexu...|Investigation com...|
|            Burglary|Investigation com...|
|Violence and sexu...|Status update una...|
|Violence and sexu...|Status update una...|
+--------------------+--------------------+
```

Figure 2.7.3 Crime type and last outcome of a specific location

And the result is displayed in a tabular form. For Leicestershire on the location 'On or near A607', the last outcome of Crime type 'Burglary', 'Other theft' and 'vehicle crime' is obtained as 'Investigation complete; no suspect identified'.

Similarly, for Northumbria, on the location 'On or near A19' different Crime types have different last outcomes. For example, Crime type- Anti-social behaviour has no last outcome. Likewise for public order, the last outcome is 'Status update unavailable ' for the respective location.

2.8) ANALYSIS BY COMBINING TWO DATA SETS

The method union() is used to merge two data sets into a single data set. It performs the same functions as the union operation. It is an in-built function defined as the set that contains all of the elements from both sets that are not duplicates. Here the two datasets of Leicestershire and Northumbria are combined to form new data frame called 'Leis_North' and analysis is done. The following Figure 2.8.1 provides the query and printed schema of the union.

```
In [56]: #combining of two data sets into single dataset
         Leis_North=df_leicestershire.union(df_northumbria1)

In [57]: Leis_North.printSchema()

root
 |-- Crime ID: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- Reported by: string (nullable = true)
 |-- Falls within: string (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Location: string (nullable = true)
 |-- LSOA code: string (nullable = true)
 |-- LSOA name: string (nullable = true)
 |-- Crime type: string (nullable = true)
 |-- Last outcome category: string (nullable = true)
 |-- Context: string (nullable = true)
```

Figure 2.8.1 Union data

Now all the queries and analysis performed earlier is also carried out for this new data set.

```
In [58]: # to return first four rows
         Leis_North.take(4)

Out[58]: [Row(Crime ID=None, Month='2021-03', Reported by='Leicestershire Police', Falls within='Leicestershire Police', Longitude=-1.21
         0015, Latitude=52.62141, Location='On or near Hobill Close', LSOA code='E01025631', LSOA name='Blaby 002A', Crime type='Anti-so
         cial behaviour', Last outcome category=None, Context=None),
          Row(Crime ID='d6aa170f22d346877589643f030399e22cf990385fb7b7bfa7758510a35a9f8d', Month='2021-03', Reported by='Leicestershire
         Police', Falls within='Leicestershire Police', Longitude=-1.214176, Latitude=52.621663, Location='On or near Lowland Avenue', L
         SOA code='E01025631', LSOA name='Blaby 002A', Crime type='Burglary', Last outcome category='Investigation complete; no suspect
         identified', Context=None),
          Row(Crime ID='f35db016e317df79fe1b7410bc316ae5b80ef4dd59257cc15cb28576a314707a', Month='2021-03', Reported by='Leicestershire
         Police', Falls within='Leicestershire Police', Longitude=-1.21556, Latitude=52.619443, Location='On or near Lancelot Close', LS
         OA code='E01025631', LSOA name='Blaby 002A', Crime type='Other theft', Last outcome category='Investigation complete; no suspec
         t identified', Context=None),
          Row(Crime ID='ac17ff0d0dbdbfdc48cda0fe36fe215c2e46efcad63d4e40dabd2a9436de7f57', Month='2021-03', Reported by='Leicestershire
         Police', Falls within='Leicestershire Police', Longitude=-1.212828, Latitude=52.622715, Location='On or near Kings Drive', LSOA
         code='E01025631', LSOA name='Blaby 002A', Crime type='Other theft', Last outcome category='Investigation complete; no suspect i
         dentified', Context=None)]
```

Figure 2.8.2 take() function for Leis_North

```
In [59]: #to find the count of dataset
         Leis_North.count()

Out[59]: 26264
```

Figure 2.8.3 count() for union data

In addition to this, the total count of the data set is taken using 'count()' function which is 26264.

```
In [61]: # to rename the column names
         Leis_North1=Leis_North.withColumnRenamed("Crime ID","Crime_ID")
         Leis_North2=Leis_North1.withColumnRenamed("Month","Month_March")
         Leis_North3=Leis_North2.withColumnRenamed("Reported by","Reported_by")
         Leis_North4=Leis_North3.withColumnRenamed("Falls within","Falls_within")
         Leis_North5=Leis_North4.withColumnRenamed("Longitude","longitude")
         Leis_North6=Leis_North5.withColumnRenamed("Latitude","latitude")
         Leis_North7=Leis_North6.withColumnRenamed("Location","location")
         Leis_North8=Leis_North7.withColumnRenamed("LSOA code","LSOA_code")
         Leis_North9=Leis_North8.withColumnRenamed("LSOA name","LSOA_name")
         Leis_North10=Leis_North9.withColumnRenamed("Crime type","Crime_type")
         Leis_North11=Leis_North10.withColumnRenamed("Last outcome category","Last_outcome")
         Leis_North12=Leis_North11.withColumnRenamed("Context","context")

In [62]: # to show ten rows of data frame
         Leis_North12.show(10)
```

```
+--------------------+----------+--------------------+--------------------+---------+---------+--------------------+---------+
----------+--------------------+--------------------+-------+
|            Crime_ID|Month_March|         Reported_by|        Falls_within|longitude| latitude|            location|LSOA_code|
LSOA_name|          Crime_type|       Last_outcome|context|
+--------------------+----------+--------------------+--------------------+---------+---------+--------------------+---------+
----------+--------------------+--------------------+-------+
|                null|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.210015| 52.62141|On or near Hobill...|E01025631|
Blaby 002A|Anti-social behav...|                null|   null|
|d6aa170f22d346877...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.214176|52.621663|On or near Lowlan...|E01025631|
Blaby 002A|            Burglary|Investigation com...|   null|
|f35db016e317df79f...|    2021-03|Leicestershire Po...|Leicestershire Po...| -1.21556|52.619443|On or near Lancel...|E01025631|
Blaby 002A|         Other theft|Investigation com...|   null|
|ac17ff0d0dbdbfdc4...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.212828|52.622715|On or near Kings ...|E01025631|
Blaby 002A|         Other theft|Investigation com...|   null|
|de0b9fa13a7d37a83...|    2021-03|Leicestershire Po...|Leicestershire Po...| -1.21556|52.619443|On or near Lancel...|E01025631|
Blaby 002A|         Other theft|Investigation com...|   null|
|e1d9a2447154862b7...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.214176|52.621663|On or near Lowlan...|E01025631|
Blaby 002A|Violence and sexu...|Investigation com...|   null|
|cbef62e2c348e7892...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.210015| 52.62141|On or near Hobill...|E01025631|
Blaby 002A|Violence and sexu...|Unable to prosecu...|   null|
|                null|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.225664|52.616057|On or near Yew Close|E01025632|
Blaby 002B|Anti-social behav...|                null|   null|
|3fd7d94146f2ddfad...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.217867|52.618424|On or near Excali...|E01025632|
Blaby 002B|Criminal damage a...|Investigation com...|   null|
|83a20fd73aedc87f6...|    2021-03|Leicestershire Po...|Leicestershire Po...|-1.216845|52.619442|On or near Merlin...|E01025632|
Blaby 002B|         Other theft|Investigation com...|   null|
+--------------------+----------+--------------------+--------------------+---------+---------+--------------------+---------+
----------+--------------------+--------------------+-------+
only showing top 10 rows
```

Figure 2.8.4 Leis_North12- Renamed dataframe

In the following step the data frame 'Leis_North12 ' is registered as table using SQL context and the table name becomes 'Leis_North2021'. This table is created by selecting only the columns of interest.

```
In [63]: #to register DataFrame as table
         sqlContext.registerDataFrameAsTable(Leis_North12,"Leis_North_2021")

In [64]: sqlContext.sql("select * from Leis_North_2021")

Out[64]: DataFrame[Crime_ID: string, Month_March: string, Reported_by: string, Falls_within: string, longitude: double, latitude: doubl
         e, location: string, LSOA_code: string, LSOA_name: string, Crime_type: string, Last_outcome: string, context: string]

In [65]: sqlContext.sql("select Reported_by, Falls_within, longitude, latitude, location,\
                          Crime_type, Last_outcome from Leis_North_2021 group\
                         by Reported_by, Falls_within, longitude, latitude, location,\
                         Crime_type, Last_outcome").show()
```

Reported_by	Falls_within	longitude	latitude	location	Crime_type	Last_outcome
Leicestershire Po...	Leicestershire Po...	-1.196626	52.618462	On or near Royce ...	Vehicle crime	Investigation com...
Leicestershire Po...	Leicestershire Po...	-1.20127	52.773874	On or near Cradoc...	Drugs	Status update una...
Leicestershire Po...	Leicestershire Po...	-1.220907	52.773249	On or near Cumber...	Violence and sexu...	Investigation com...
Leicestershire Po...	Leicestershire Po...	-1.210446	52.751695	On or near Acer C...	Anti-social behav...	null
Leicestershire Po...	Leicestershire Po...	-1.223841	52.752475	On or near Parkin...	Vehicle crime	Investigation com...
Leicestershire Po...	Leicestershire Po...	-1.366325	52.536765	On or near Southf...	Theft from the pe...	Investigation com...
Leicestershire Po...	Leicestershire Po...	-1.159307	52.658981	On or near Oronsa...	Violence and sexu...	Investigation com...
Leicestershire Po...	Leicestershire Po...	-1.148629	52.655354	On or near Burnha...	Violence and sexu...	Unable to prosecu...
Leicestershire Po...	Leicestershire Po...	-1.142734	52.634769	On or near Bath Lane	Criminal damage a...	Investigation com...
Leicestershire Po...	Leicestershire Po...	-1.103079	52.646082	On or near Merewo...	Violence and sexu...	Unable to prosecu...
Leicestershire Po...	Leicestershire Po...	-1.065197	52.64676	On or near Winslo...	Drugs	Local resolution
Leicestershire Po...	Leicestershire Po...	-1.068449	52.639819	On or near Ocean ...	Violence and sexu...	Investigation com...
Leicestershire Po...	Leicestershire Po...	-1.105228	52.635661	On or near St Sav...	Anti-social behav...	null
Leicestershire Po...	Leicestershire Po...	-1.090615	52.641303	On or near Traffo...	Violence and sexu...	Unable to prosecu...
Leicestershire Po...	Leicestershire Po...	-1.15266	52.63191	On or near Hinckl...	Public order	Unable to prosecu...
Leicestershire Po...	Leicestershire Po...	-1.05894	52.622878	On or near Newhav...	Anti-social behav...	null
Leicestershire Po...	Leicestershire Po...	-1.071311	52.624666	On or near Ensbur...	Violence and sexu...	Unable to prosecu...
Leicestershire Po...	Leicestershire Po...	-1.168136	52.624531	On or near Thurli...	Violence and sexu...	Unable to prosecu...
Leicestershire Po...	Leicestershire Po...	-1.12231	52.616451	On or near Adderl...	Other theft	Unable to prosecu...
Leicestershire Po...	Leicestershire Po...	-1.050974	52.641505	On or near Kirkwa...	Public order	Offender given a ...

only showing top 20 rows

Figure 2.8.5 SQL table of union data

Now, to retrieve summary data based on one or more groups, 'groupby()' clause is used. Here the columns Crime type and Last outcome are grouped and count() is acquired. And then the column 'count' is sorted using 'sort()' in ascending order.

```
In [66]: #to find the count of union of two specific columns
         Leis_North12.groupby("Crime_type","Last_outcome").count().sort("count").show()
```

Crime_type	Last_outcome	count
Burglary	Suspect charged a...	1
Shoplifting	Further action is...	1
Other theft	Further investiga...	1
Bicycle theft	Further investiga...	1
Other theft	Further action is...	1
Drugs	Offender given pe...	1
Shoplifting	Awaiting court ou...	1
Vehicle crime	Awaiting court ou...	1
Violence and sexu...	Offender given pe...	1
Possession of wea...	Further investiga...	1
Vehicle crime	Offender given a ...	1
Theft from the pe...	Court result unav...	2
Robbery	Local resolution	2
Vehicle crime	Further investiga...	2
Burglary	Offender given a ...	2
Burglary	Awaiting court ou...	2
Other theft	Action to be take...	2
Other crime	Awaiting court ou...	2
Vehicle crime	Local resolution	3
Other theft	Awaiting court ou...	3

only showing top 20 rows

Figure 2.8.6 sort() for union data

For the first row, count is 1 when the Crime type is Burglary and Last outcome is ‘suspect charged as a part of another case’. Similarly the rest of them can be inferred.

```
In [67]: #to count of the null values
         i = 0 # reset counter
         #get the total records amount
         total = Leis_North.count()

         print("Total Records = " + str(total))

         #print the amount of columns
         print("Total columns = " + str(len(Leis_North.columns)))
         print("----------------------------------------")

         #loop entire table and get the missing value number and missing rate of each column
         while i < len(Leis_North.columns): #loop through all columns
             print(str(i+1) + "." + str(Leis_North[i]))
             print("  Missing Values = ")
             print("----------------------------------------")
             mv = Leis_North.select([count(when(Leis_North[i].isNull(),\
                                                True))]).show() #check for missing values
             i = i+1 #counter add 1
```

```
Total Records = 26264
Total columns = 12
----------------------------------------
1.Column<b'Crime ID'>
  Missing Values =
----------------------------------------
+-------------------------------------------------+
|count(CASE WHEN (Crime ID IS NULL) THEN true END)|
+-------------------------------------------------+
|                                             7748|
+-------------------------------------------------+

2.Column<b'Month'>
  Missing Values =
----------------------------------------
+-----------------------------------------+
|count(CASE WHEN (Month IS NULL) THEN true END)|
+-----------------------------------------+
|                                        0|
```

Figure 2.8.7 Count of the null values

As earlier, the column ‘context’ has the highest missing values. Similarly the summary of the data frame can be obtained as follows,

```
In [68]: #to find count,mean,stddev,min,max
         #reset counter
         i=0

         print("----------------------------------------")

         #loop each column and get the descriptionof each one
         while i < len(Leis_North.columns):
             mv = Leis_North.where(Leis_North[i] =='').count() #get the missing value number of current column
             print(str(i+1) + "." + str(Leis_North[i]))
             Leis_North.where(Leis_North[i] !='').describe(Leis_North.columns[i]).show()
             print("----------------------------------------")
             i = i+1 #counter add 1
```

```
----------------------------------------
1.Column<b'Crime ID'>
+-------+--------------------+
|summary|            Crime ID|
+-------+--------------------+
|  count|               18516|
|   mean|                null|
| stddev|                null|
|    min|000025d45f1ff2db8...|
|    max|fffec74c7bd120b5c...|
+-------+--------------------+

----------------------------------------
2.Column<b'Month'>
+-------+-------+
|summary|  Month|
+-------+-------+
|  count|  26264|
|   mean|   null|
```

Figure 2.8.8 Summary of union data

```
In [69]: #to find count of values of a specific column
         sqlContext.sql("select Crime_type, count(*) as No_of_Crime_type from\
                        Leis_North_2021 group by Crime_type order by Crime_type").show()
```

```
+--------------------+----------------+
|          Crime_type|No_of_Crime_type|
+--------------------+----------------+
|Anti-social behav...|            7748|
|       Bicycle theft|             205|
|            Burglary|             762|
|Criminal damage a...|            2413|
|               Drugs|             621|
|         Other crime|             601|
|         Other theft|            1391|
|Possession of wea...|             179|
|        Public order|            2369|
|             Robbery|             101|
|         Shoplifting|             892|
|Theft from the pe...|             103|
|       Vehicle crime|             918|
|Violence and sexu...|            7961|
+--------------------+----------------+
```

Figure 2.8.9 Count of Crime type of union data

Here the count of the column ‘Crime type’ is calculated. It should be noted that for union data, the highest count is for ‘Violence and sexual offences’ which is 7961. But Robbery has the least count 101.

EXPLORATORY ANALYSIS- INSIGHTS AND DISCOVERIES

Exploratory data analysis can aid in the identification and discovery of evident errors and outliers in datasets, the study of correlation, finding the patterns in the data and the development of new insights. Here 'Pixiedust' which is an open source Python library, is imported to visualise the data. Pixiedust display includes scatter plots, maps, bar charts, line charts etc. It enables the insightful representation of data and eases the users to interpret and arrive at conclusions.

```
In [74]: import pixiedust
```

```
Pixiedust database opened successfully

Pixiedust version 1.1.15

Warning: You are not running the latest version of PixieDust. Current is 1.1.15, Latest is 1.1.19

Please copy and run the following command in a new cell to upgrade: !pip install --user --upgrade pixiedust

Please restart kernel after upgrading.
```

```
In [75]: display(leicestershire_202112)

In [76]: display(northumbria1_202112)

In [77]: display(Leis_North12)
```

Figure 3.1 import and display() in Pixiedust

To start with the visualisation, the pie charts of the variable ‘Crime type’ is portrayed using the function ‘display()’

Here the pictorial representation of the Crime type of Leicestershire, Northumbria and union data (Leis_North) is also obtained. By analysing this, the frequently occurring Crime type in Leicestershire is ‘Violence and sexual offences’ which is 40%. And the Crime type ‘ theft from the person’ has the lowest percentage. In the case of Northumbria, the most common happened crime is ‘Anti-social behaviour’ and the least occurred crime is ‘Robbery’. And finally when these data frames are combined, both the Crime types ‘Violence and sexual offences’ and ‘Anti-social behaviour’ equally have a higher percentage. Here it is inferred that the proportion of ‘Violence and sexual offences’ in Leicestershire and ‘Anti-social behaviour’ obtained separately is equal to that proportion acquired when both datasets are in Northumbria. These results are the same as that of the count obtained using SQL queries.

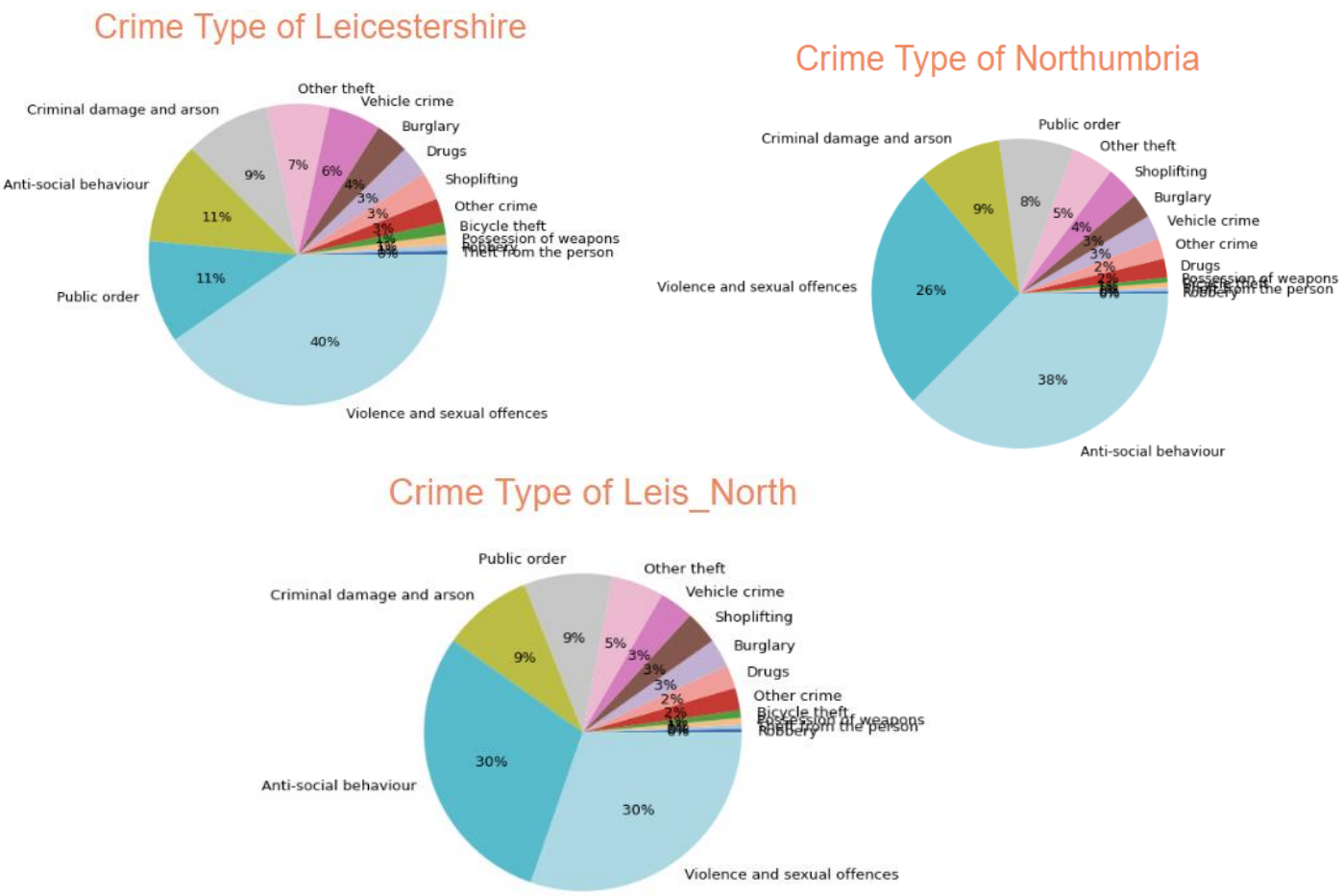


Figure 3.3 Pie chart for Crime type

The above shown visualisation can be done in an alternative way. In the next step, the map of the Crime types along with latitudes and longitudes is pictured below.

Figure 3.4 depicts the location and count of the Crime type ‘Violence and sexual offences’ in Leicestershire. The count of values is indicated by different colour spots.

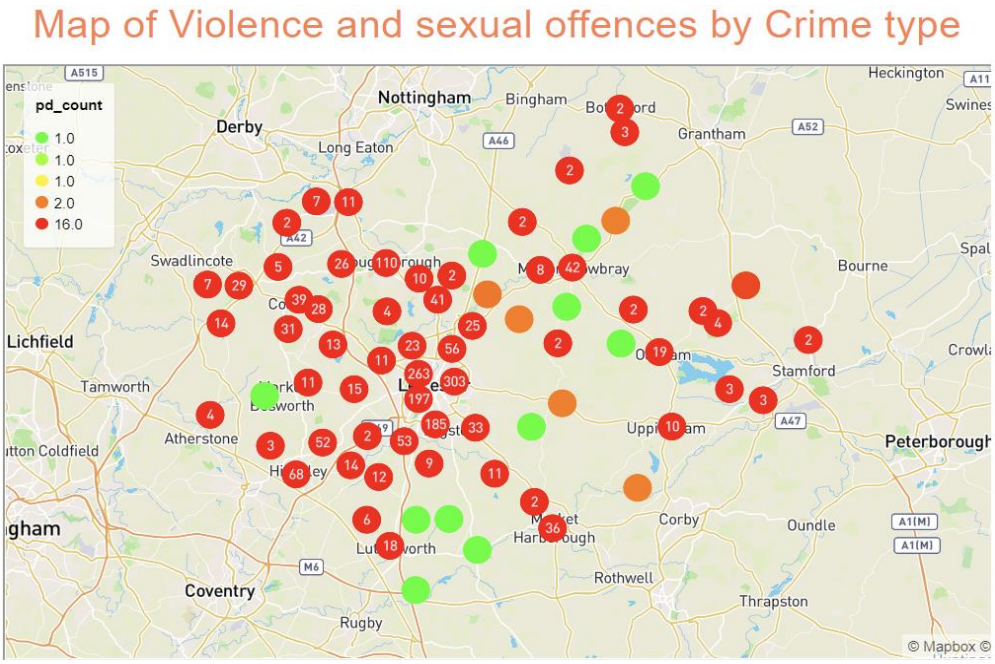


Figure 3.4 Map of Leicestershire

Likewise, the overview of Crime type 'Anti-social behaviour' mapped using Pixiedust as shown in the Figure 3.5.

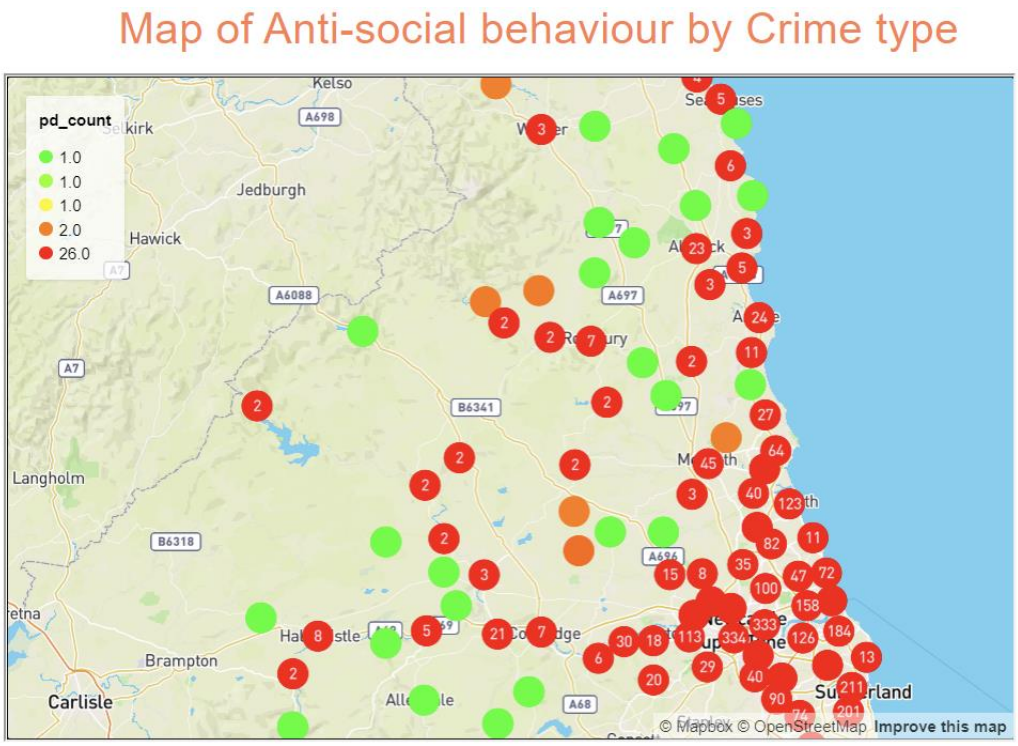
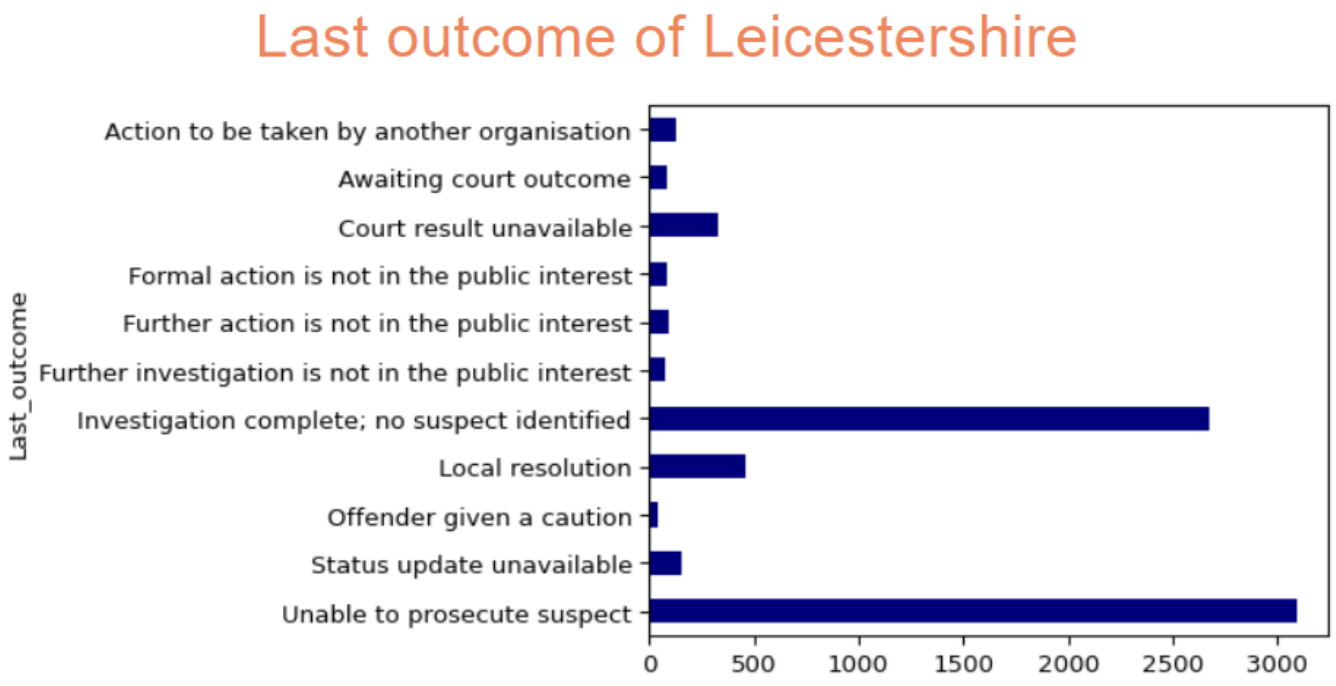


Figure 3.5 Map of Northumbria

Clustered areas implies the concentrated rate of the specific crime. By zooming in the map of Leicestershire there is a case of 'Violence and sexual offences' reported near to De Montfort University. And in the map of Northumbria, a case of 'Anti-social behaviour' is reported near Northumbria University City Campus East. So precautions should be taken near the institutions by hiring new officers or assigning existing officers in ways that put them on the street in larger numbers or for longer periods of time. Also all crimes are not alike

and cannot be weighted equally. It is difficult to develop and implement a crime prevention strategy that addresses everything, even all crime within a general category.

Furthermore, the bar graph of the variable 'Last outcome' of both places is illustrated in the Figure 3.6



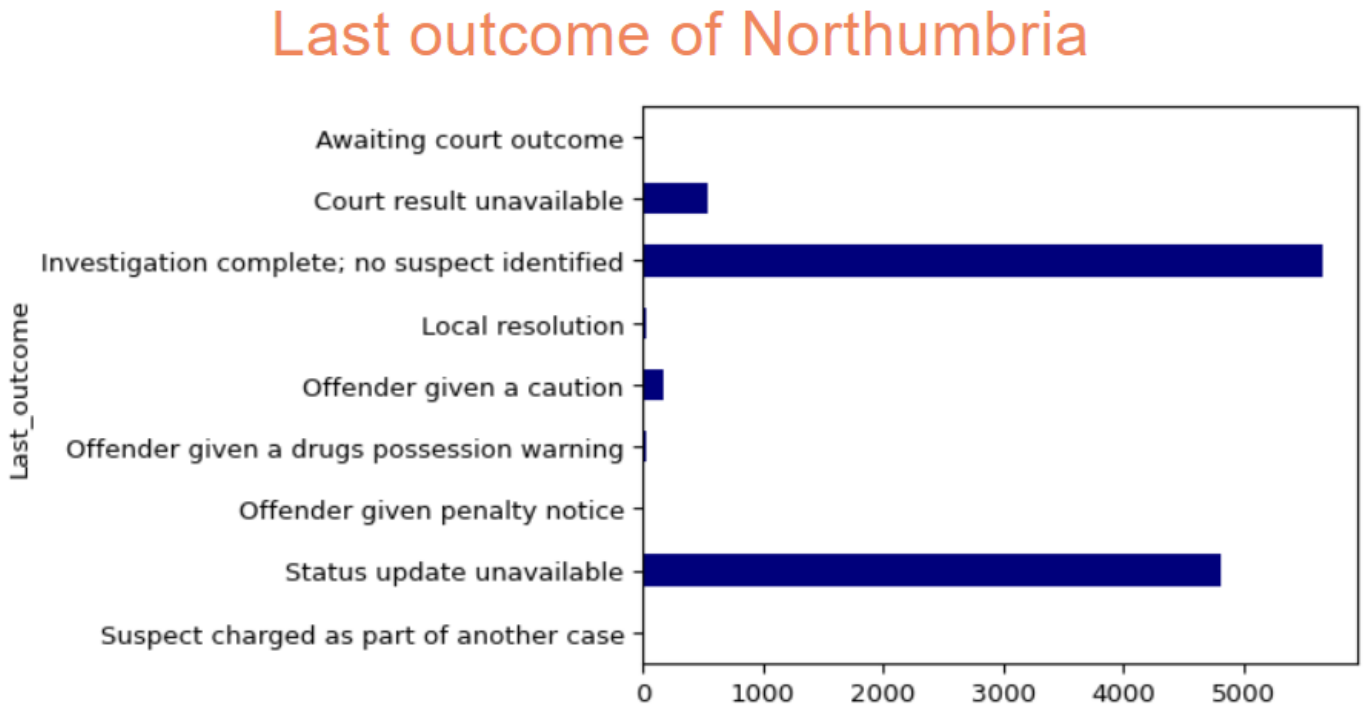


Figure 3.6 Bar graphs of three datasets

In the case of Leicestershire, most of the cases have the last outcome 'Unable to prosecute suspect' and for Northumbria, the maximum number of cases results in the completion of the investigation but no suspect identified; these results convey that more efficient investigation has to be done regarding the cases. Even if the union of data is visualised, ' investigation complete; no suspect identified' is the highest range 'Last outcome'.

CONCLUSION AND FUTURE WORK

Crime is one of the major challenges faced by the Government where innocent citizens become the victims in criminal activities across the world. In this report, descriptive and exploratory analysis of two data sets '2021-03-leicestershire-street.csv' and '2021-03-northumbria-street.csv' is carried out. The given data sets are cleaned and transformed and the required analysis is done in Jupyter Notebook using Apache Spark to reach at new insights and trend prediction. For that, essential queries are executed and the results are visualised through graphs.

Here the inferences are made concentrating on the variables 'Location', Crime type' and 'Last outcome'. Speaking of the intensity of Crime type in both places, the population density of Leicestershire is around 561,635 and the highest Crime type is 'Violence and sexual offences'. On the other hand, in the case of Northumbria, in an approximate population of 322,434, 'Anti-social behaviour' is the most common Crime type. From this observation, it is understood that, even though Northumbria has the lowest population density compared to Leicestershire, it possesses the excess crime rate. Both of them consist of wide range of other

crimes and this is to be taken into account seriously. Since data sets are from two places, the genetic, cultural and environmental factors also influence in committing crimes.

Also considering the COVID-19 scenario, 2021 was the peak time of break out where all the people were locked up in their houses. Most of them lost their jobs and suffered from the financial crisis. So this might also influence the crime rate of two places

Now, speaking of the sample size, the sample of Northumbria is greater than Leicestershire and it might imply that it contains unreported crimes. And moreover, it is desirable to include quantitative values and other variables like age band, Gender, history of suspect, Location type, population, different time frame can be included for future studies. Literally, the ultimate aim of this crime analysis is examine the crime rate to ensure morale and safety of society.