Home Office

Seasonal Variance of Crime and Police Efficiency

Author:
Kazuma Kato

Group:
November 2018

January 4th, 2019

Abstract

Seasonal variance in the efficiency of the Metropolitan and City of London police departments were investigated in conjunction with criminal activity. It was found that police efficiency decreases whilst criminal activity increase over the hotter months. A monthly crime forecast model for the Greater London boroughs was built using the analyses of this investigation, returning an error margin of 5.52%.

Contents

1 Introduction **2**

2 Modelling **3**

3 Data Sourcing **5**

4 Data Considerations **5**
4.1 Normalisation . . . 5
4.2 Limitations of the Data . . . 6
4.3 Introducing Measures . . . 7
4.4 Time Delay / Lag . . . 8

5 Engineering **9**
5.1 Initialising the Database . . . 9
5.2 Geographic Dimensions . . . 11
5.3 Creating a Date Key using a Recursive CTE . . . 11
5.4 Transforming Population Data into a Dimensional Table . . . 12
5.5 Combining Weather Tables into DimWeather . . . 12
5.6 Preparing and Creating Crime Dimension, DimCrime . . . 12
5.7 Cleaning City of London Police Data . . . 13
5.8 Populating DimCrime & FactCrime with City Data . . . 13
5.9 Cleaning Metropolitan Police Data . . . 14
5.10 Populating DimCrime & FactCrime with City Data . . . 14
5.11 Evaluating the Data Warehouse Architecture . . . 15

6 Analysis **16**
6.1 Preparing the Database for Analysis : Police Efficiency . . . 17
6.2 Preparing the Database for Analysis : Crime Trends . . . 17
6.3 Analysing Police Efficiency . . . 18
6.4 Analysing Crime Trends . . . 20

7 Forecasting Crime **23**
7.1 Creating a View to Export Forecast Model . . . 23
7.2 Testing the Forecast Model . . . 25

8 Conclusion **26**

9 Next Steps **26**

Appendix of Code **27**

1 Introduction

The efficiency of the Metropolitan and City of London police dapartments, in tandem with criminal activity levels, in relation to meteorological variance is investigated in this report.

A dashboard for Police Efficiency across London (in comparison to weather) has been built, as well as a forecasting dashboard which predicts a subsequent month's increase or decrease in criminal activity, per borough of London (based on a supplied weather forecast).

By sourcing data on recorded crimes and meteorological figures during the period of 2012 - 2016, it was possible to analyse the statistical significance between weather and varying levels of police efficiency and criminal activity.

It was found that the police forces governing the Greater London region has a negative correlation with 'good' weather conditions - where 'good' weather is defined by high temperature, low rainfall, more sunlight hours. That is to say, the police officers are less efficient during the summer months.

It was also found that criminal activity has a positive correlation with 'good' weather coniditions. That is to say, more crimes are committed during the summer months. Percentage changes between weather and crime was found to have a Pearson product-moment correlation coefficient of 0.81, and relational coefficient (gradient) of 0.753.

The equation of change was found to be:

$$\Delta Crime(\%) = 0.753 * \Delta Weather(\%) \tag{1}$$

A forecasting model was built in order to predict changes in criminal activity, per borough of London. This model takes an input of the preceeding month's crime statistics and the weather forecast of the target month. Testing this model was accurate to 5.52%.

The Git Repository can be found ***here***

2 Modelling

Given the two analytical concepts which we wish to bring together, we must ensure that the architecture can accomodate both.

We must first consider the format of data sets available: a list of all individual crimes, complete with details, and weather data broken down into $5000m^2$ grids. As such, let us define the metrics of the two key models which we wish to investigate:

1. Police Efficiency
   1.1. Number of crimes committed in each borough, broken down by month
   1.2. Proportion of those crimes which resulted in only a warning being issued
   1.3. Proportion of those crimes which resulted in a penalty being issued
   1.4. Meteorological data of each respective borough, for the month in which the crime was committed
2. Crime Trends
   2.1. Proportion of crimes committed in each borough in relation to its population, broken down by both month and crime category
   2.2. Meteorological data of each respective borough, for the month in which the crime was committed, broken down by weather stat temperature, rainfall, sunlight hours

With the target in mind, we can now define the relational database model and star schema structure which we must build. The core of the databse will be a Fact table, FactCrime, which will house the core Dimensions; Crime ID, Date, Geography, Population, Weather.

The Crime ID will be supplemented in a Dimension which contains all the information about the crime itself, Geography with informations, etc.. The population and weather dimensions will have a surrogate key introduced, since they are dependent on both geography and time. Dictionaries will be introduced to the crime dimension since the qualitative information is most often not required when querying; making the majority of queries more efficient.

The resulting Databse Structure can be explored on the next page. It will be shown in the Engineering section that the database was indeed constructed in this way.

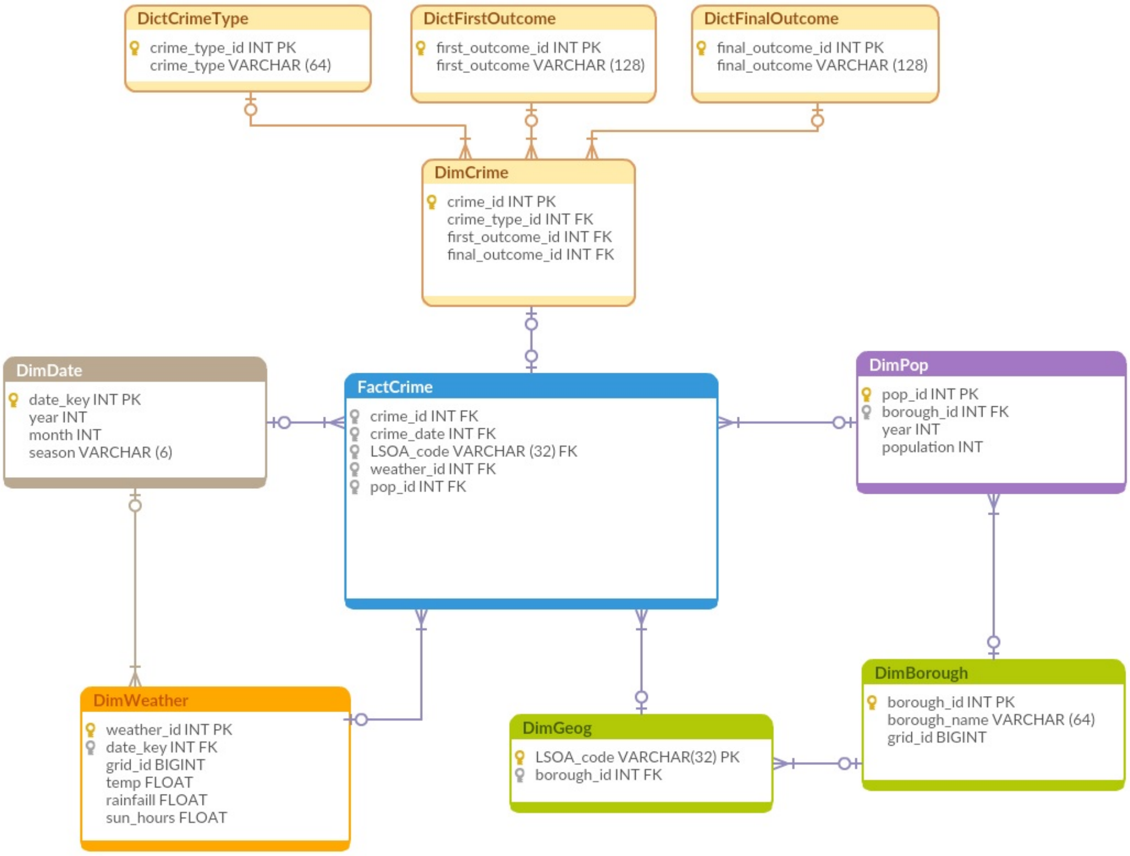

Figure 1: Datbase Diagram

A non-clustered index is created on LSOA code as this is the strongest linking factor in the FactCrime table, speeding up queries and increasing efficiency.

From this database structure, we can create the necessary queries and views which will be used in the Analysis section.

3 Data Sourcing

There are Four primary data sets which are required for this investigation:

1. Crime Data (reported - 2012 - 2016 and outcome - 2012-2018)
   1.1. Source: *Police Data Website*
2. LSOA Atlas, linking to Boroughs of London (2011 census)
   2.1. Source: *London Data Gov Website*
3. London Borough Population by Year (2012 - 2016)
   3.1. Source: *London Data Gov Website*
4. Meteorological Data across Greater London, broken down by $5000m^2$ grid (UKCP09 Dataset 2012 - 2016)
   4.1. Source: *Met office, held by CEDA (Centre for Environmental Data Analysis*

The Crime Data is the core of the model, and will be enriched by the other three sources and enable us to perform the appropriate analyses. All data is obtained in CSV format from a reliable source, however, it is prudent to check the data's format and consistency before importing it. A sense check shows that the data is already clean.

4 Data Considerations

We must first consider the types of challenges which we might face, as well as any adjustments which may be a prerequisite to overcoming these challenges. This section looks at identifying the possible difficulties and explore the necessary corrections where applicable.

4.1 Normalisation

Any information is alsmost meaningless without a reference point of sorts. As an example - there is no conclusion to be made about the number of burglaries observed in two particular areas if we do not know some other information which links them:

1. Physical area covered
2. Number of houses
3. Population density

Which are relevant in the context of burglaries, but not so much in the relation to pickpocketing.

As such, we must be very careful in our attempt to draw any conclusions without any method of normalising the data. At a basic level - a simple "per population density (1000/sq mile)" would be an acceptable example in the case above.

In the case of testing police efficiency, simply choosing a normalised measure - *proportion of crimes which result in warnings or penalties* - is intself an act of normalisation.

When observing crime trends in conjunction to weather, we must normalise the number of crimes commited per borough by their respective populations. Thus, giving a *number of crimes committed per 1000 population* measure - a measure which has significance.

4.2 Limitations of the Data

It is also critical for us to understand the limitations of the Data available to us, such as the level of completeness, coherence/consistency as all of these may not be available to us in its entirety.

As such, it is important to draw limits of acceptability. For example, we have chosen not to pursue the Metropolitan Police data on Stop and Searches since a third of that data does not include the location; making it impossible to allocate to a borough, thus introducing (literally) incredible uncertainty.

The movement of people must also be taken into consideration. For example, Hyde Park will see a high number of tourists while Winter Wonderland is occupying the park. There are a lot of visitors in the area, however, that does not imply that we can assume that they contribute to the local population. City of London and Westminster are of relative significance here.

The applicability of any conclusion only applies to the Greater London area, and is not to be extrapolated to the rest of the UK without solid evidence.

4.3 Introducing Measures

When attempting to measure the effectiveness of policing in the boroughs of London, great care must be taken when creating a scale by which comparisons will be made.

To begin with, when the number of police warnings issued is compared to the number reported crimes is introduced as a measure, we must also define the meaning (or interpretation) of the spectrum.

A high relative proportion of warnings being issued can be understood to indicate more anti-social behaviour in a given month and also a decline in police efficiency.

A lower relative proportion of penalties being issued can be interpreted as two possibilities:

1. Given a **low** proportion of warnings; a raise in anti-social crimes (mostly civilians can be blamed). Therefore police are being hindered by the activity of a particular crime type.
2. Given the **same or similar** proportion of warnings; the two are not correlated and we must look to find a link to another factor - namely weather. We can conclude that the police force is *inefficient*
3. Given a **high** proportion of warnings; police force is shown to be *efficient*

The proportion of penalties issued and warnings issued must be evaluated together, as will be seen in the analysis section.

4.4 Time Delay / Lag

When looking at static data of a dynamic variable, it is imperative to consider the difference in time between an event and its outcome. In this particular case, we must realise that crimes are not always (half of the time not) issued an 'outcome' at the same time that they are reported. It takes time for the police department as well as the judicial system to issue a verdict once an arrest/investigation has been made.

In order to account for this consideration, we will have to evaluate the average time difference (per type of crime) for a crime to be processed and an outcome issued. This is subtley done in the engineering section, and a dataset of 2012-2016 chosen specifcally to account for this lag.

In the instance of multiple charges for a single crime ID, we will look at the first instance of a charge being issued (we will not consider participants who have not been charged for any specific crime). Further details are investigated in the engineering section.

5 Engineering

This section follows the architecture of the database as laid out in the modelling section; following the chronological order of engineering whilst maintaining a continuous narrative. All references to small data sources held in the Git repository is denoted by (Git) identifier.

All references to code can be found in the Appendix of Code, however, any reference in the narrative contains an embedded hyperlink to the corresponding code at the end of this document - denoted by (CXX). For reading convenience, each corresponding code snippet also contains a 'back to section' hyperlink which returns the reader to the corresponding section of the document which directed them there.

5.1 Initialising the Database

After creating the databse (C1) and core schemas, the core crime data can be imported (into the stage schema) using Alteryx (see Figure 2).

Since the source directory structure spans across a multitude of folders, but only contains 4 types of files, it is a trivial task for Alteryx. The City of London Police Data (referred to as simply City throughout the code) and the Metropolitan Police Data (referred to as Met) is divided into "street" and "outcome" data, linked solely on "crime ID".

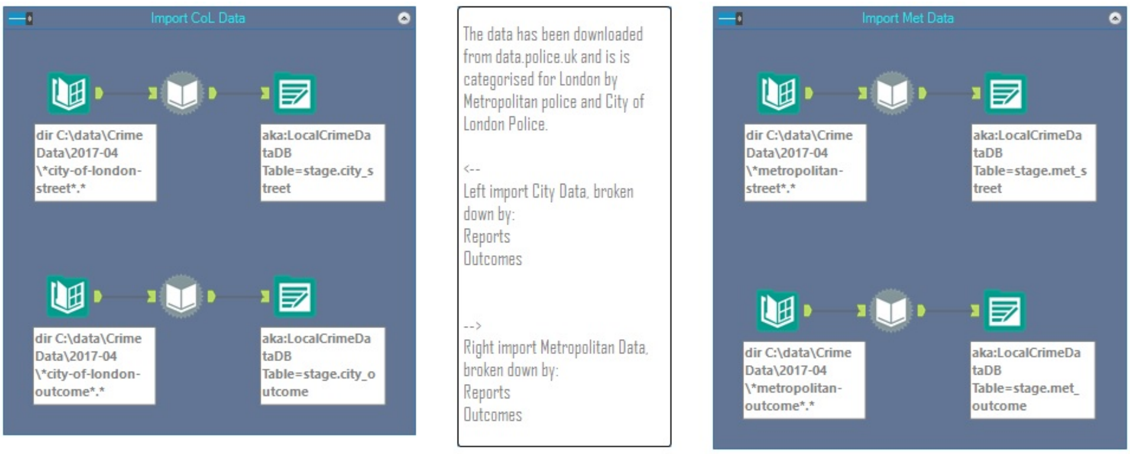

Figure 2: Import All Crime Data via Alteryx

As such, the four tasks simply search for all files with their respective file path identifiers, such as **city_street** or **met_outcome** and plece them in their respective *stage.table_name*, e.g. - *stage.city_outcome*.

The Weather Data can also be imported using Alteryx, see Figure 3.
Following the flow of the diagram, the data is all in one file, but contains temperature, rainfall and sunlight hours data vertically, and the grid id horizontally. As such, we are required to unpivot the data into a format which gives us the grid id vertically.

This is easily achieved using the Alteryx pivot tool. To further distill the data into respective weather dimension (stat = sun/rain/temp), a basic filter is used. The syntax is identified using the native Alteryx dialogue. The result is the three tables: stage.temp, stage.rain, stage.sun.

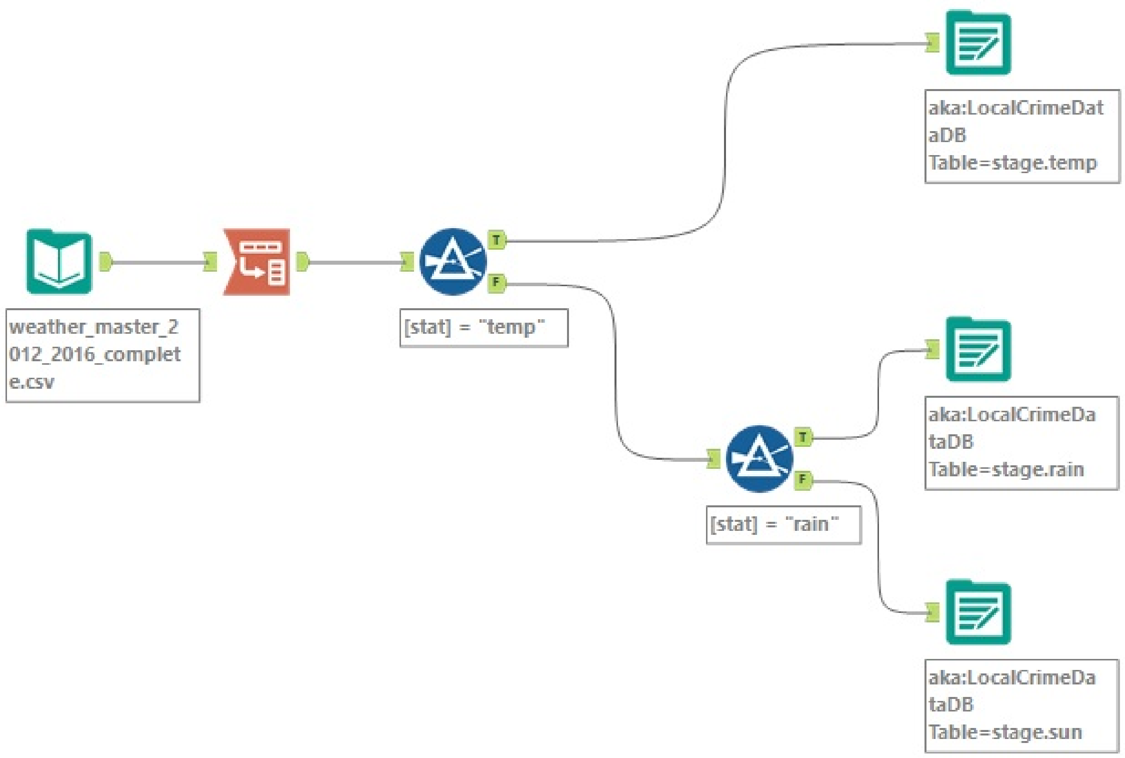

Figure 3: Import Weather Data via Alteryx

A brief investigation (C2) into the nature of the data which we are working with reveals to us that there are often multiple charges for a given crime_id, where multiple perpetrators are involved in a single crime. Some further eyeballing of the data shows us that in such a case with multiple outcome entries for a single crime entry; the first entry of outcome almost always is the definitive entry - i.e. there are no cases where the first outcome entry results in no penalty and proceeding entries result in a penalty.

This is most useful information when it comes to deduplicating the data later on as we can ignore any secondary entries in the outcome table.

It is also easy to see that all entries where the crime type is "anti-social behaviour" does not have a crime id. As such, we will generate unique when dealing with the DimCrime and FactCrime tables later.

5.2 Geographic Dimensions

We begin by importing the Borough Data (Git) from the data.gov website and creating the Dimborough (C4). A quick investigation of the maximum string length enables us to decide the VARCHAR length to use for the table.

From this, we can create DimGeog (C3), as this links the LSOA code of the crime table with the borough. We can cleverly use the stage.borough table in order to pull the borough_id codes using a join.

5.3 Creating a Date Key using a Recursive CTE

Since we have a dataset with a non-standard date formay, YYYY-MM, and we also require a surrogate key for our DimDate table - it would be prudent to use this format to generate our own datekey. We can do so by using a recursive CTE (C5) and casting a concatenated substring as the datekey. Encapsulating the result in a table with an assigned primary key gives us the Date Dimension, DimDate.

5.4 Transforming Population Data into a Dimensional Table

Similarly to the datekey, we will need to generate a population id surrogate key when inserting into the DimPop table. We will begin by defining the table which we want to fill (C6) and also a temporary table with the unpivoted data.

This allows us to easily add the pop_id column in conjuction with the temptable into DimPop. The borough id is taken from the DimBorough table via an inner join.

5.5 Combining Weather Tables into DimWeather

This section is slightly more complicated in that the weather data is divided between three separate tables; each containing a different stat values, but housing the same range of date (year) and grid name.

As such, we will begin by setting up the required table with keys (C7) and staging the stat tables into three respective temptables; each with the data cleansed into the correct format. Finally, bringing this all together as a final combined temptable and generating our desired surrogate key, weather_key, as a concatenation of both date_key and grid_id. A sense check confirms that this process was successful so that we may insert the result into the DimWeather table.

5.6 Preparing and Creating Crime Dimension, DimCrime

Before we can create the DimCrime table, we must first create the dictionaries by identifying the distinct crime types, first and final outcomes.

The dictionaries are easily constructed (C8) with a simple SELECT DISTINCT query into a preformed table (with an INT identity key). A zero row is forcibly inserted by temporarily allowing an identity insert to account for NULL values - for reported crimes which do not have a recorded outcome; e.g. - in the case of Anti-social crimes having no outcome data.

With the dictionaries set up, we can then proceed to create the DimCrime table (C9) as well as creating the FactCrime table (C10)

5.7 Cleaning City of London Police Data

Before we can populate either DimCrime or FactCrime, we must first clean both the 'street' and 'outcome' data. This will be achieved using a two-stage process. First, we will prepare the stage.city_street and stage.city_outcome tables (C11) into stage.city_street2 and stage.street_outcome2 respectively, deduping crime_ids and generating unique crime IDs for 'Anti-social behaviour' type crimes in the process. We do this using the ROW_NUMBER() and NEWID() functions.

5.8 Populating DimCrime & FactCrime with City Data

This two-stage process begins by creating a comprehensive temptable using two CTEs (C12), which is then used to populate the DimCrime and FactCrime tables (from the same temptable).

The temptable accounts for NULL outcomes, extracts dates, re-casts extracted dates as a date_key and also provides a sense check for the number of months between the date of a particular given crime and its outcome date. This is important, as outlined in the modelling section, since we have specifcally chosen a 2012-2016 dataset in order to counteract the lag between report and outcome.

A brief eyeballing and MAX check of the months_between field shows us that we were correct to encapsulate 2 years worth of additional outcome data - beyond the recorded crime data!

Using this new temptable, it is now a very straight-forward process to insert the relevant fields into both DimCrime (C13) and FactCrime (C14). Care must be taken to use a LEFT JOIN when joining on outcome data, as we must allow NULLs in these columns. Conversely, we must make sure to use an INNER JOIN when joining with DimGeog, as we specifically want to *exclude* any data for crimes occuring in LSOA areas *outside* of the Greater London boundary. A sense check proves our pertinence worthwhile.

5.9 Cleaning Metropolitan Police Data

We will employ the same method for cleaning the Met street data (C15) and outcome data (C16) of cleaning as we did when cleaning the City data; deduping crime_ids and generating new ids for those with crime type 'Anti-social behaviour'.

5.10 Populating DimCrime & FactCrime with City Data

By using a two-stage process via the use of a temporary table which combines both street and outcome Met data (C17), we can employ the same methods as we did for the City data to further populate the DimCrime (C18) and FactCrime (C19)tables - completing our crime data!

A sense check proves useful here when checking the difference in imported rows between the Dimensional table and the Fact table. By the construction of the joins used in the process, we can assume that this difference is due to crimes with LSOA codes outside of the Greater London region. Selecting these isolated crime ids and checking them on a map quickly reveals that this is indeed true.

5.11 Evaluating the Data Warehouse Architecture

As was laid out in the Modelling section, the aim was to create a star schema structure. By generating a database diagram within the MS SQL Server Management Studio, we can confirm that the architecural schematics have been followed, and that the Data Warehouse has been created successfully, complete with all primary-foreign key relationships. See Figure 4 below.

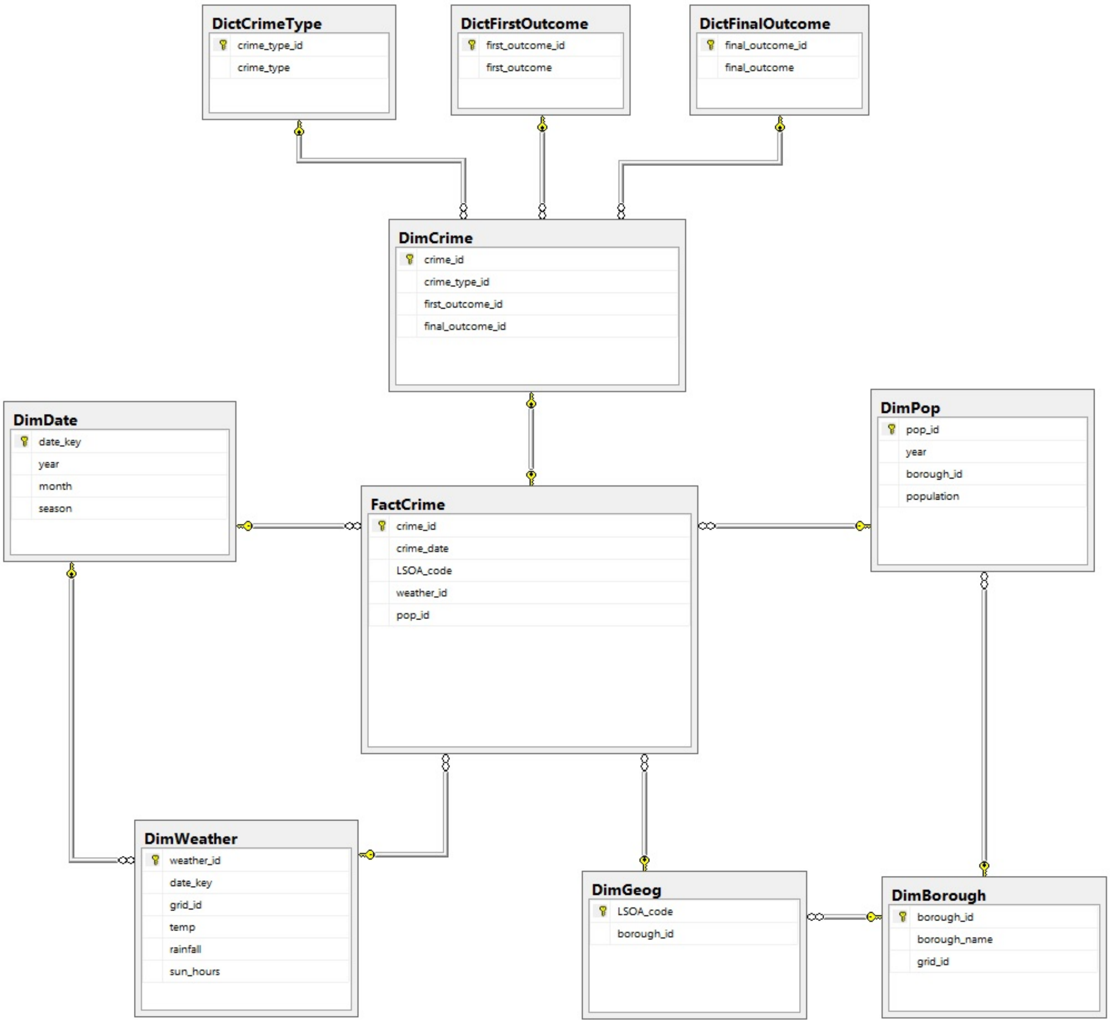

Figure 4: Server Management Studio Database Diagram

6 Analysis

This section is split into three main components.

1. Query our database so that we may extract the data in a format which can be visualised easily using Tableau.
2. Visualise our police efficiency metrics to test our hypothesis, ultimately create a dashboard.
3. Enrich our crime data with weather data in order to identify any meteorological patterns.

We will analyse the police efficiency metrics: ratio of warnings to crimes; ratio of outcomes to crimes as outlined in the hypothesis. By visualising the data, it becomes clear there is indeed a seasonal pattern to police efficiency within Greater London. Despite controversial predictions, we demonstrate that the Metropolitan Police force is less efficient during the summer months, with good weather.

An cartographical representation of this analysis is encorporated within the dashboard, making it easy to identify how each individual borough of London performs on an efficiency scale.

Breaking down the crimes committed across Greater London by borough, category and contributing weather stat will clearly demonstrate a meteorological pattern to criminal activity also. Conversely to a reduction in police efficiency during the summer months, there is a clear rise in criminal activity during the same months.
All data is normalised using geo-dependent population (and also for the relevant time period). Furthermore, we will be focusing on the *percentage difference; month-on-month* as is critical when observing changes in weather.

Since our data is both normalised and self-contained, the combined findings of a reduce in police efficiency and increase in criminal activity during the summer months can be independently attributed to the weather - *and not to be mis-concluded that police are less efficient due to an increase in criminal activity.*

6.1 Preparing the Database for Analysis : Police Efficiency

In order to make our queries more efficient, we will begin by creating two temporary dictionaries (C20) in order to classify the types of outcomes of a crime:

1. No Consequence
2. Penalty Issued
3. Warning

This will also make things much easier when constructing our higher level query.

By combining our dictionaries in a query together (C21) with COUNT() and CASE statements, we can simultaneously extract outcome categories from both the first_outcome and final_outcome columns. This gives us the correct outcome category without having to worry about "which column it was in".

Using the above in a CTE will enable us to evaluate the ratios, per borough, of the proportion of warnings and penalty outcomes in comparison with the crime committed.

6.2 Preparing the Database for Analysis : Crime Trends

code.22aCreating a dictionary (C22) in a similar method enables us to identify 4 primary crime categories:

1. Theft
2. Violent
3. Anti-social
4. Other

Which we will use to construct an export table (C23) from which we can evaluate the weather stats by borough. A quick sense check shows us that the correct number of rows have been returned.

Since we also wish to look at the general correlation between crime category and weather stat, we will aso create an unpivoted view (C24). This allows us to visualise in tableau as a stacked view.

We must also break down the correlation trends by category of time as well as weather stat (a 3x4 grid). More importantly, the only important factor which we want to evaluate is the percentage change from the previous month! With some well-constructed CTEs, we will create a view (C25) which can be visualised in Tableau.

6.3 Analysing Police Efficiency

With the above data easily brought to life via Tableau, we can immediately see a seasonal pattern based on meterological variance. Below is a dashboard, also hosted online (linked image to interactive web host).

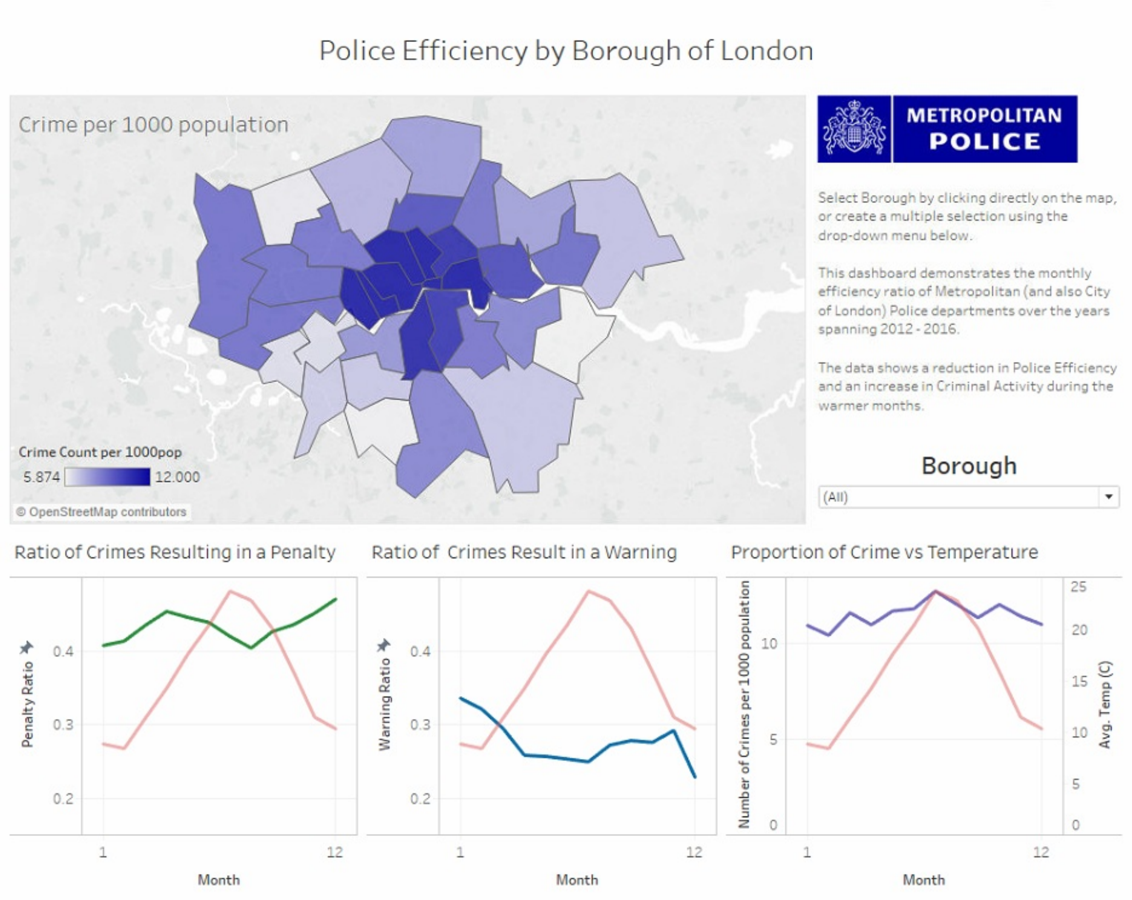

Figure 5: Police Efficiency Across Greater London

As the dashboard shows, there does indeed seem to be a seasonal pattern for all three graphs.

Firstly, the Ratio of Crimes which result in a penalty being issued has two peaks and two troughs. It does not demand a great leap of imagination to understand a rough cause for these.

1. January has a less than ideal ratio of crime rates due to a larger than average proportion of anti-social behaviour - New Year's binge drinking spike can attribute to a large proportion of this. Recall that these recorded crimes have no outcomes, and result only in warnings or local resolutions.
2. April sees a peak in prosecution, which can largely be caused by the Easter Holiday period, where the population of London temporarily decreases as families and professionals go on holiday.
3. December also sees an increase in prosecution efficiency for similar reasons as Easter - people tend to go back to their 'roots' outside of London or go overseas for the festive period. One may also be inclined to guess that the 'holiday spirit' could also be a contributing factor (a social construct which may play some relevance here).
4. The warmer months clearly see a decline in police efficiency, with a larger proportion of crimes commited which result in no action being taken against criminals.

It is important to take both the proportion of warnings and penalties together - especially in regard to the fourth point enumerated above. That is, that it is indeed a case of police *inefficiency* which plays the causal factor here - and ***not*** due to a surge in anti-social crime (which would have no penalty by default, and only result in a warning).

We can say this in confidence due to the relative *fall* in the number of warnings. This is not only contraversial evidence, but should represent a critical factor in police operations and budgeting with very wide-bearing implications.

6.4 Analysing Crime Trends

Having looked at the correlation between police efficiency and meteorological conditions, one would also expect to see a relationship between criminal activity and weather changes. This section of the investigation keeps in mind that the aim is to create a forecast model, and so measurements and comparisons are made quantitatively.

We will begin by exploring the different categories of crime, their monthly variance and their trend in comparison to meteorological changes. Reviewing the data which we prepared for visualisation in an earlier section yields an interesting result - the absolute values for the relationship between the weather only appears to have a similar trend to the number of anti-social crimes committed. This is demonstrated in the graph below:

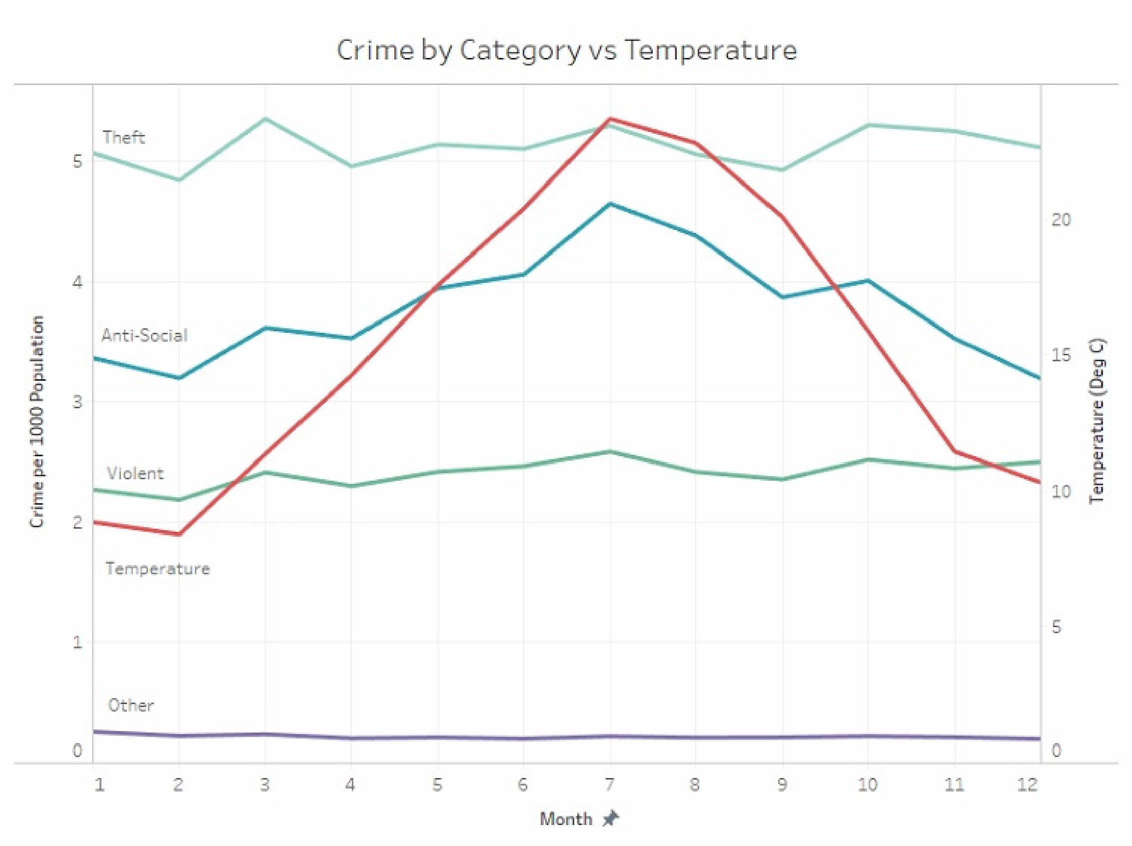

Figure 6: Crimes Committed (per 1000 pop) vs Temperature, by Category and Month

This initially seems to hold only a moderate amount of significance, however, it is absolutely critical to recall that the only important factor in respect to weather (for our case, at least) is the *change* in weather. As humans, our limited capacity only allows us to evaluate a *relative* change in weather - i.e. "it feels a bit colder today compared to yesterday".

As such, we must evaluate the data accordingly; by representing the data in terms of percentage changes in weather, which will give us the correct method of normalisation required for us to assess the data accuracately. Doing so yields the following graphs, broken down by category of crime and attributing weather stat:

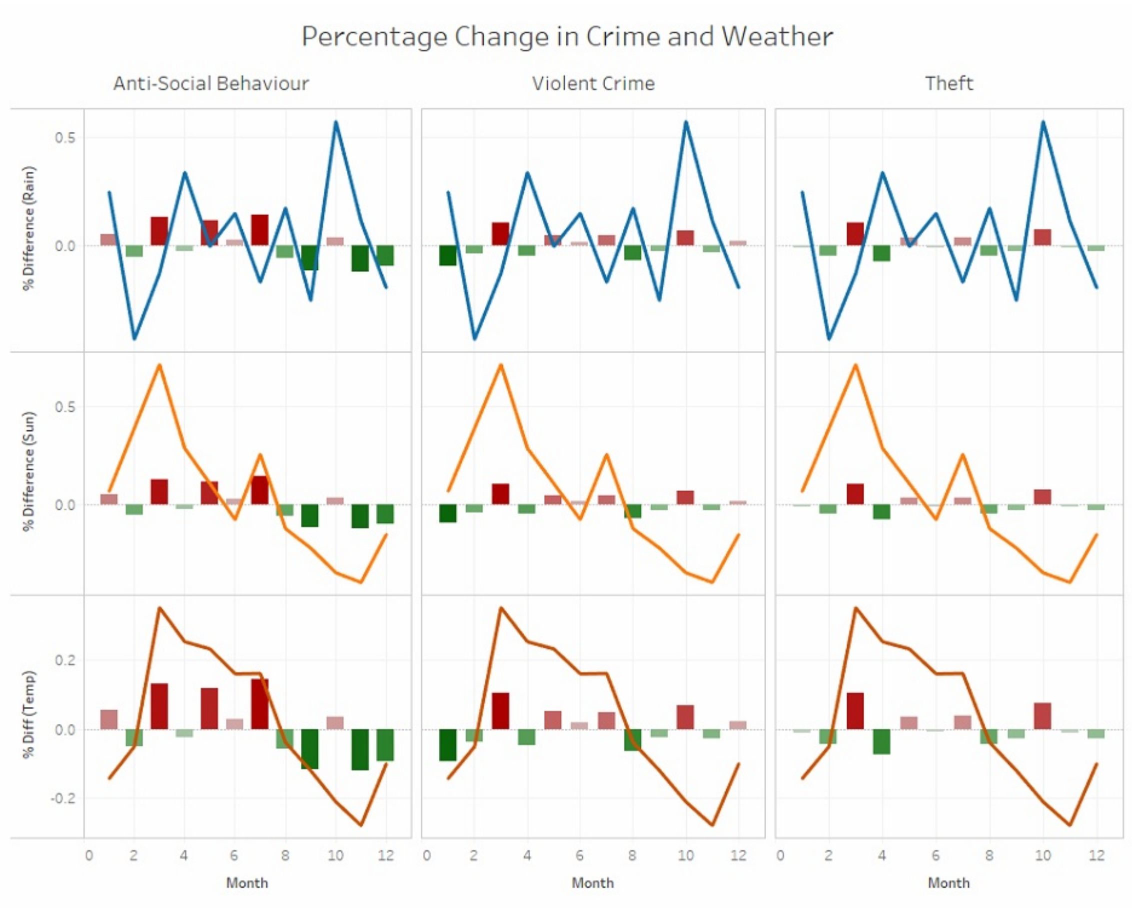

Figure 7: Percentage Change in Crime vs Percentage Change in Weather

Inspecting the graph series carefully reveals that there does indeed appear to be a function of the three weather stats which contributes to the change in crime.

That is to say:

$$f(\Delta crime) = f(\Delta temperature, \Delta rainfall, \Delta sunlighthours) \tag{2}$$

It is difficult to evaluate such a function using tools within the scope of this analysis, however, that will not stop us from attempting to make an approximation, albeit a mathematically crude one.

By eyeballing the source data of the above graph, we can see some rough patterns:

1. Temperature seems to be most correlated to crime changes.
2. When temperature appears to be inversely related, we see spikes of rainfall - indicating that above a certain threshold, rainfall is a dominating factor - i.e. criminals will put up with a small amount of rain, but will reconsider their motivation during periods of heavy rainfall.
3. A decrease in the number of sunlight hours also seems to be a deterrent to most criminals - also implying that more sunlight hours increases criminal activity.
4. Easter time and Christmas time is anomolous in that the relative population of London decreases during these periods - as mentioned in the preceeding section.

We can use this information, combined with a little more investigation in order to construct a basic forecasting model; covered in the next section.

7 Forecasting Crime

As our analysis shows, we have certainly found a decent level of correlation between the change in weather and the change in criminal activity.

We can combine these three factors as in equation (2) above by simply finding the mean percentage change between the three months, per borough, per month. We can then evaluate this attempt by using a Pearson Product-Moment Correlation Coefficient - which will tell us the strength of our correlation. A value of +1 indicates a perfect correlation; 0 indicates no correlation and -1 indicates a perfect negative correlation.

$$r = \frac{N\sum XY - (\sum X \sum Y)}{\sqrt{(N\sum x^2 - (\sum x)^2)(N\sum y^2 - (\sum y)^2)}} \tag{3}$$

This yields a surprisingly high correlation coefficent of 0.81, which indicates a confident level of correlation. A quick check of percentage difference in crime and percentage difference in average weather change (average of the three stats) yields a gradient of 0.753. That is to say that the crime percentage difference is equal to three quarters of the weather percentage change.

That is to say that the equation of change of crime in relation to weather is given by:

$$\Delta Crime(\%) = 0.753 * \Delta Weather(\%) \tag{4}$$

7.1 Creating a View to Export Forecast Model

Using these this powerful information, we can now formulate an SQL query (C26) (to forecast 'next month's crime' and visualise this in Tableau. We will create this model using a series of CTEs, combining the crime data of June-2016 and the weather data of July-2017 (let us pretend that the July meteorological data is a 'forecast'). The resulting table must then be reconverted into a count of crime (unnormalised) (C27) and then unpivoted and exported as a view (C28).

With this view, we can construct a dashboard in Tableau which predicts the additional crime for the proceeding month (can be positive or negative for a decrease). The dashboard is interactive, and individual (or a multiple) selection of boroughs can be evaluated. As this particular example shows, the Eastern boroughs (which will experience different weather) will experience a decrease in crime, whilst the Western boroughs expect a rise in crime. This is visualised on the next page, Figure 8 (linked image to interactive web host).

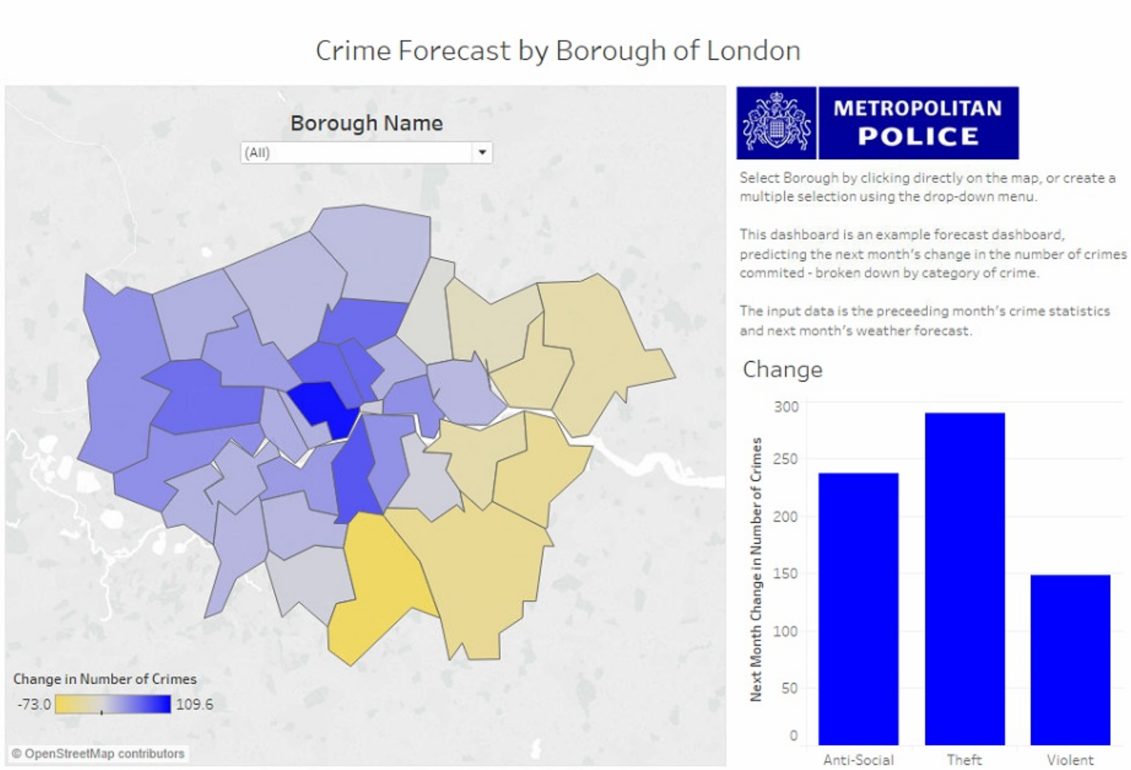

Figure 8: Crime Forecast by Borough of London

7.2 Testing the Forecast Model

By taking the known recorded crime statistics of July 2016, we are able to test our 'forecasted' crime figures and compare (test) them against the real figures.

Since the objective is simply to test an isolated instance, it is permissible to use a series of temptables in order to manipulate the data into a format which we can evaluate (C29).

Evaluating the inaccuracy (percentage difference between the predicted figure and actual figure) for each borough will return the level of inaccuracy of our model overall. Finding the mean value across all boroughs yields a total inaccuracy of -5.52%. Our Forecast has proven, in this test, to be very successful, underestimating the number of crimes committed in July 2016 by only 5.52%. This certainly indicates that exploring the relationship between weather and crime would prove worthwhile.

8 Conclusion

This investigation set out to explore the relationship between police efficiency and criminal activity and meteorological variation, specifically within the Greater London region.

It was found that during police inefficiency is negatively correlated with 'good' weather - where 'good' is defined by high temperature, more sunlight hours and less rainfall. That is to say, the Metropolitan police force is less efficient during periods of good weather.

Furthermore, this analysis was conducted with normalised data, thus establishing police efficiency is independent of anti-social behaviour caused by criminals. As such, the decrease in efficiency is correctly attributed to weather, and not what would otherwise be attributed to 'civilian nuisance'.

In addition, it was also found that changes in criminal activity are correlated with meteorological changes. That is to say, criminal activity increases with 'good weather'.

By leveraging the learnings of the analysis, we were able to construct a Crime Forecast Model which was able to successfully predict a test case with an impressively low error margin of 5.52%.

9 Next Steps

The investigation has shown that the correlation between criminal activity and weather is strong enough to merit further investigation.

By using the learnings of the forecast model, it would be pertinent to explore a machine learning model in order to more dynamically forecast changes in crime to a more granular level - by LSOA code (not just borough) and to a daily basis.

Such a model would be able to help boost police efficiency and develop a more targeted distribution of police officers and help optimise budget allocation.

The findings of this investigation, although somewhat contraversial, warrant further investigations into the causal relationship between weather and police efficiency. Such questions could be combined with an employee engagement study in order to maximise the potential of such an investigation.

Appendix of Code

1 Create Database 28
2 Initial Investigation 28
3 Create DimGeog 29
4 Create DimBorough 30
5 Create DimDate 31
6 Create DimPop 33
7 Create DimWeather 35
8 Create Dictionaries 38
9 Create DimCrime 40
10 Create FactCrime 41
11 Clean City Data 42
12 Set up City2 Data in temptable 43
13 Use temptable to insert City Data into DimCrime 44
14 Use temptable to insert City Data into FactCrime 45
15 Clean Met Street Data 46
16 Clean Met Outcome Data 47
17 Set up Met2 Data into temptable 47
18 Use temptable to insert Met Data into DimCrime 49
19 Use temptable to insert Met Data into FactCrime 49
20 Create Outcome Type Temporary Dictionaries 50
21 Evaluate Warning and Outcome Ratios; Insert into Export Table . . . 52
22 Create Crime Type Dictionary 54
23 Create Weather Trend Export Table 54
24 Create Unpivoted Weather Trend View 56
25 Create Crime View : Seasonal Variance by Crime Type 57
26 Create Crime Forecast Export Table 59
27 Intermediate Crime Count Forecast 60
28 Create View : Crime Change Forecast 62
29 Test Forecast Model 62

Create Database and Schemas

```
CREATE DATABASE CrimeData
GO

        CREATE SCHEMA stage
        GO

        CREATE SCHEMA tableau
        GO

        CREATE SCHEMA trash
        GO
```

Listing 1: Create Database

Back to Section (of the above code)
Initial Investigation

```
-- check to see what the data looks like
;WITH cte
AS
(
SELECT
    cs.[Crime ID]
   ,COUNT(cs.[crime id]) AS countcrime
FROM stage.city_street AS cs -- COUNT = 21957
LEFT JOIN stage.city_outcome AS co
  ON cs.[Crime ID] = co.[Crime ID]
GROUP BY cs.[Crime ID]
)
SELECT
    *
FROM cte
WHERE countcrime != 1 -- this is DIFFERENT from > 1


-- Let's now take a look at some random crime_id's which have
-- shown to have multiple entries

SELECT
    * -- returns 1 row
FROM stage.city_street
WHERE [Crime ID] = '006724129
    e38b131479ac24be625428afd0d6897dbf5f165f12c1a9b01840aa2'

```

```sql
SELECT
    * -- returns 3 rows
FROM stage.city_outcome
WHERE [Crime ID] = '006724129
    e38b131479ac24be625428afd0d6897dbf5f165f12c1a9b01840aa2'

-- we can see from this that the outcome table has:
--      duplicate for "suspect charged"
--      extra line for "offender given community sentence"
--    where the only difference can be found in the [Outcome Type]
```

Listing 2: Initial Investigation

Back to Section (of the above code)
Create DimGeog

```sql
-- DimGeography is imported into the stage schema as a task
-- csv imported,  change col: lsoa code to width = 10

-- clean up:
;WITH cte
AS
(
SELECT
    Codes As LSOA_code
   ,REPLACE(names,RIGHT(names,5),'') AS borough_name
FROM stage.geog
)

SELECT
    CAST(LSOA_code AS VARCHAR(10)) AS LSOA_code
   ,CAST(b.borough_id AS TINYINT) AS borough_id
INTO dbo.DimGeog
FROM cte AS c
INNER JOIN stage.borough AS b
   ON b.borough_name = c.borough_name

-- create PK
ALTER TABLE DimGeog
ALTER COLUMN LSOA_code VARCHAR(10)NOT NULL
GO

ALTER TABLE DimGeog
ADD CONSTRAINT pk_lsoa_code PRIMARY KEY (LSOA_code)
GO

-- we can add FK AFTER DimBorough is created...
```

```
ALTER TABLE dbo.dimgeog
ADD CONSTRAINT fk_geog_borough_id FOREIGN KEY (borough_id)
  REFERENCES dimborough (borough_id)
GO
```

Listing 3: Create DimGeog

Back to Section (of the above code)
Create DimBorough

```
-- import as task, then process:
SELECT
    CAST(borough_id AS TINYINT) AS borough_id
   ,CAST(borough_name AS VARCHAR (22)) AS borough_name
   ,grid_id
INTO dbo.DimBorough
FROM stage.borough

--------------------------------------
          -- this gives us the required VARCHAR length
          SELECT
            MAX(LEN(borough_name))
          FROM stage.borough
--------------------------------------

-- create keys:

ALTER TABLE dbo.DimBorough
ALTER COLUMN borough_id TINYINT NOT NULL
GO

ALTER TABLE dbo.dimborough
ADD CONSTRAINT pk_borough PRIMARY KEY CLUSTERED (borough_id)
GO

ALTER TABLE dbo.dimborough
ALTER COLUMN grid_id BIGINT NOT NULL
GO
```

Listing 4: Create DimBorough

Back to Section (of the above code)

Create DimDate

```
-- We need to generate the date_key column! Let's do the recursive cte
-- loop -> dump it into a #temptable and then substring out the
-- datekey, making sure it is an INT.

-- First, make sure the #temptable is clear..
IF OBJECT_ID('tempdb..#temptable') IS NOT NULL
  BEGIN
    DROP TABLE #temptable
  END

;WITH cte
AS
(
SELECT
    CAST('2012-01-01' AS DATE) AS dt
   ,DATEPART(yy, CAST('2012-01-01' AS DATE)) AS Yr
   ,DATEPART(mm, CAST('2012-01-01' AS DATE)) AS Mnth
UNION ALL
SELECT
    DATEADD(mm,1,dt) AS dt
   ,DATEPART(yy, DATEADD(mm,1,dt)) AS Yr
   ,DATEPART(mm, DATEADD(mm,1,dt)) AS Qtr
FROM cte
WHERE DATEADD(mm,1,dt) < GETDATE()
)
SELECT
    *
INTO #temptable
FROM cte
OPTION (MAXRECURSION 0)


-- great! Now we need to generate the date_key from this

;WITH ctemp
AS
(
SELECT
    dt
   ,yr
   ,mnth
   ,CAST(CONCAT(SUBSTRING(CAST(dt AS VARCHAR(4)),1,4)
          ,SUBSTRING(CAST(dt AS VARCHAR(7)),6,2)) AS INT) AS date_key
FROM #temptable
)
SELECT
```

```
    date_key
   ,yr AS [year]
   ,mnth AS [month]
   ,CASE
     WHEN mnth IN (3,4,5) THEN 'spr'
     WHEN mnth IN (6,7,8) THEN 'sum'
     WHEN mnth IN (9,10,11) THEN 'aut'
     ELSE 'win'
    END AS season
INTO dbo.DimDate
FROM ctemp

-- MAKE SURE KEY IS SET

ALTER TABLE dbo.DimDate
ALTER COLUMN date_key INT NOT NULL
GO

ALTER TABLE dbo.DimDate
ADD CONSTRAINT pk_date PRIMARY KEY CLUSTERED (date_key)
GO
```

Listing 5: Create DimDate

Back to Section (of the above code)

Create DimPop

```
-- Here, we will have to first unpivot before populating the table.
-- Begin by setting up the table:

CREATE TABLE dbo.DimPop
  (
   pop_id INT PRIMARY KEY
  ,[year] SMALLINT NOT NULL
  ,borough_id TINYINT NOT NULL -- FK in DimBorough
  ,[population] INT NOT NULL
  )
GO

-- Set FK
ALTER TABLE DimPop
ADD CONSTRAINT fk_borough_id FOREIGN KEY (borough_id)
  REFERENCES DimBorough (borough_id)
GO

-- Unpivot into a temptable so that it's in the correct format:
IF OBJECT_ID('tempdb..#temppop') IS NOT NULL
  BEGIN
    DROP TABLE #temppop
  END

SELECT
   LTRIM(RTRIM(bor_name)) AS bor_name
  ,[year]
  ,[population]
INTO #temppop
FROM
(
SELECT
   bor_name
  ,pop_2012
  ,pop_2013
  ,pop_2014
  ,pop_2015
  ,pop_2016
FROM stage.pop
) AS unp
UNPIVOT
(
  [population]
  FOR [year]
    IN (
        [pop_2012]
```

```
        ,[pop_2013]
        ,[pop_2014]
        ,[pop_2015]
        ,[pop_2016]
          )

) AS upp

-- now throw it into the DimPop with correct key.
INSERT INTO DimPop
   (
    pop_id
   ,[year]
   ,borough_id
   ,[population]
   )
SELECT
    CAST(CONCAT(RIGHT([year],4), borough_id) AS INT) AS pop_id
   ,CAST(RIGHT([year],4) AS SMALLINT) AS [year]
   ,borough_id
   ,[population]
FROM #temppop AS t
INNER JOIN DimBorough AS b
  ON b.borough_name = t.bor_name
```

Listing 6: Create DimPop

Back to Section (of the above code)

Create DimWeather

```sql
-- Set up the table which we want to insert into:
CREATE TABLE DimWeather
  (
   weather_id BIGINT PRIMARY KEY
  ,date_key INT
  ,grid_id BIGINT
  ,temp FLOAT
  ,rainfall FLOAT
  ,sun_hours FLOAT
  )
GO

-- Set key
ALTER TABLE DimWeather
ADD CONSTRAINT fk_weather_date_key FOREIGN KEY (date_key)
  REFERENCES DiMDate (date_key)
GO

-- Nowe we will set up 3 temptables - for {temperature(temp), rainfall, sun_hours}
-- Begin with standard #temptable clear.
IF OBJECT_ID('tempdb..#temptemp') IS NOT NULL
  BEGIN
    DROP TABLE #temptemp -- (temporary table for temperature)
    DROP TABLE #temprain
    DROP TABLE #tempsun
    DROP TABLE #temptotal
  END -- we can do this all at once, since we will never have only an isolated instance!


-- create the temptables in the same order as above, then mash them into the DimWeather

------*********** make #temptemp ***********------
;WITH ctetemp
AS
(
SELECT
   CAST(CONCAT(SUBSTRING(CAST([year] AS VARCHAR(4)),1,4)
         ,SUBSTRING(CAST([year] AS VARCHAR(7)),6,2)) AS INT) AS date_key
  ,Name AS grid_id
  ,CAST(Value AS FLOAT) as value
FROM stage.temp
)
SELECT
```

```
    CAST(CONCAT(date_key, grid_id) AS BIGINT) AS weather_id
   ,value AS temp
INTO #temptemp
FROM ctetemp

------*********** make #temprain **********------
;WITH cterain
AS
(
SELECT
    CAST(CONCAT(SUBSTRING(CAST([year] AS VARCHAR(4)),1,4)
          ,SUBSTRING(CAST([year] AS VARCHAR(7)),6,2)) AS INT) AS date_key
   ,Name AS grid_id
   ,CAST(Value AS FLOAT) as value
FROM stage.rain
)
SELECT
    CAST(CONCAT(date_key, grid_id) AS BIGINT) AS weather_id
   ,value AS rainfall
INTO #temprain
FROM cterain

------*********** make #tempsun **********------
;WITH ctesun
AS
(
SELECT
    CAST(CONCAT(SUBSTRING(CAST([year] AS VARCHAR(4)),1,4)
          ,SUBSTRING(CAST([year] AS VARCHAR(7)),6,2)) AS INT) AS date_key
   ,Name AS grid_id
   ,CAST(Value AS FLOAT) as value
FROM stage.sun
)
SELECT
    CAST(CONCAT(date_key, grid_id) AS BIGINT) AS weather_id
   ,value AS sun_hours
INTO #tempsun
FROM ctesun


------*********** bring together in #temptotal **********------
;WITH ctebig
AS
(
SELECT DISTINCT
    CAST(CONCAT(d.date_key,b.grid_id) AS BIGINT) AS wid
   ,date_key
   ,grid_id
FROM DimDate AS d
```

```
CROSS JOIN DimBorough AS b -- RETURNS 2436 ROWS! (Because it goes up to
    2018)
)
SELECT DISTINCT
    wid AS weather_id
   ,date_key
   ,grid_id
   ,temp
   ,rainfall
   ,sun_hours
INTO #temptotal -- so we can just throw this into the DimWeather table
    easily
FROM ctebig AS c
INNER JOIN #temptemp AS t
   ON t.weather_id = c.wid
INNER JOIN #temprain AS r
   ON r.weather_id = c.wid
INNER JOIN #tempsun AS s
   ON s.weather_id = c.wid

     -- it returns 1740 rows, instead of 11700 in #temptemp
     -- this is okay, because it only gives the grid_id's which
     -- we already have in DimBorough
     -- 696 difference = 29 grid_id's for 24 months
     -- PERFECT!!!

-- We use #temptotal just to ensure a smooth entry into
-- the DimWeather table. (we are using cte)

INSERT INTO DimWeather
   (
    weather_id
   ,date_key
   ,grid_id
   ,temp
   ,rainfall
   ,sun_hours
   )
SELECT
    *
FROM #temptotal

-- check success, eyeball new table shows good result
```

Listing 7: Create DimWeather

Back to Section (of the above code)

Create Dictionaries

```
-- We want to create DimCrime, but we want our Database to be efficient
-- Hence we create dictionaries.
-- The #temptable below references AJKDAOPSIKJDPOANSD:PAKN

-- Begin with Crime Type:
CREATE TABLE DictCrimeType
  (
   crime_type_id INT IDENTITY PRIMARY KEY
  ,crime_type VARCHAR(50)
  )
GO

INSERT INTO dictcrimetype
  (
   crime_type
  )
SELECT DISTINCT
   crime_type
FROM #temptable

SELECT * FROM DictCrimeType

-- next; first_outcome

CREATE TABLE DictFirstOutcome
  (
   first_outcome_id INT IDENTITY PRIMARY KEY
  ,first_outcome VARCHAR(256)
  )
GO

INSERT INTO DictFirstOutcome
  (
   first_outcome
  )
SELECT DISTINCT
   first_outcome
FROM #temptable

-- force zero insert
SET IDENTITY_INSERT DictFirstOutcome ON
GO
INSERT INTO DictFirstOutcome
  (
   first_outcome_id
  ,first_outcome
```

```sql
    )
VALUES
    (0, 'None')
SET IDENTITY_INSERT DictFirstOutcome OFF
GO

--------------------------------------------------------------------------------
-- finally; final_outcome
CREATE TABLE DictFinalOutcome
    (
     final_outcome_id INT IDENTITY PRIMARY KEY
    ,final_outcome VARCHAR(256)
    )
GO

INSERT INTO DictFinalOutcome
    (
     final_outcome
    )
SELECT DISTINCT
     final_outcome
FROM #temptable

-- force zero insert
SET IDENTITY_INSERT DictFinalOutcome ON
GO
INSERT INTO DictFinalOutcome
    (
     final_outcome_id
    ,final_outcome
    )
VALUES
    (0, 'None')
SET IDENTITY_INSERT DictFinalOutcome OFF
GO
```

Listing 8: Create Dictionaries

Back to Section (of the above code)

Create DimCrime

```
CREATE TABLE dbo.DimCrime
  (
   crime_id VARCHAR(64) PRIMARY KEY
  ,crime_type_id INT
  ,first_outcome_id INT
  ,final_outcome_id INT
  )
GO

-- Set keys:
ALTER TABLE dbo.DimCrime
ADD CONSTRAINT fk_typ_id
  FOREIGN KEY (crime_type_id)
  REFERENCES dbo.DictCrimeType (crime_type_id)
GO

ALTER TABLE dbo.DimCrime
ADD CONSTRAINT fk_first_id
  FOREIGN KEY (first_outcome_id)
  REFERENCES dbo.DictFirstOutcome (first_outcome_id)
GO

ALTER TABLE dbo.DimCrime
ADD CONSTRAINT fk_final_id
  FOREIGN KEY (final_outcome_id)
  REFERENCES dbo.DictFinalOutcome (final_outcome_id)
GO
```

Listing 9: Create DimCrime

Back to Section (of the above code)

Create FactCrime

```

CREATE TABLE FactCrime
  (
   crime_id VARCHAR(64) PRIMARY KEY
  ,crime_date INT -- FK DimDate\date_key
  ,LSOA_code VARCHAR(10)
  ,weather_id BIGINT
  ,pop_id INT
  )
GO
-- make sure keys are set properly
ALTER TABLE FactCrime
ADD CONSTRAINT fk_crime_id
  FOREIGN KEY (crime_id)
  REFERENCES DimCrime (crime_id) -- OK, Done

ALTER TABLE FactCrime
ADD CONSTRAINT fk_crime_date
  FOREIGN KEY (crime_date)
  REFERENCES DimDate (date_key) -- OK, Done

ALTER TABLE FactCrime
ADD CONSTRAINT fk_LSOA_code
  FOREIGN KEY (LSOA_code)
  REFERENCES DimGeog (LSOA_code) -- OK, Done

ALTER TABLE FactCrime
ADD CONSTRAINT fk_weather_id
  FOREIGN KEY (weather_id)
  REFERENCES DimWeather (weather_id) -- OK, Done

ALTER TABLE FactCrime
ADD CONSTRAINT fk_pop_id
  FOREIGN KEY (pop_id)
  REFERENCES DimPop (pop_id) -- OK, Done
```

Listing 10: Create FactCrime

Back to Section (of the above code)

Clean City Data

```
;WITH cte1 -- start with street data
AS
(
SELECT
    CASE
     WHEN [Crime ID] IS NULL AND [Crime type] = 'Anti-social behaviour'
        THEN CAST(NEWID() AS VARCHAR (64))
     ELSE [Crime ID]
    END AS crime_id
    ,[Month]
    ,[LSOA code]
    ,[Crime type]
    ,[Last outcome category]
FROM stage.city_street
), cte2
AS
(
SELECT
    *
    ,ROW_NUMBER() OVER (PARTITION BY crime_id ORDER BY crime_id) AS rn
FROM cte1
)
SELECT
    *
INTO stage.city_street2
FROM cte2
WHERE 1=1
  AND rn = 1
  AND [LSOA code] IS NOT NULL -- WE DON'T CARE IF THERE IS NO LOCATION

;WITH cte -- then outcome data
AS(
SELECT
    *
    ,ROW_NUMBER() OVER (PARTITION BY [crime id] ORDER BY [crime id]) AS
     rn
FROM stage.city_outcome
)
SELECT
    *
INTO stage.city_outcome2
FROM cte WHERE rn = 1
```

Listing 11: Clean City Data

Back to Section (of the above code)

Set up City2 Data in temptable

```
-- We will create a pretty comprehensive #temptable here..
IF OBJECT_ID('tempdb..#temptable') IS NOT NULL
  BEGIN
    DROP TABLE #temptable
  END

;WITH cte
AS
(
SELECT
     cs.[Crime_ID] AS crime_id
    ,RIGHT(cs.[Month],2) AS crime_month
    ,LEFT(cs.[month],4) AS crime_year
    ,cs.[LSOA code] AS LSOA_code
    ,cs.[Crime type] AS crime_type
    ,ISNULL(cs.[Last outcome category], 'None') AS first_outcome
    ,ISNULL(co.[Outcome type], 'None') AS final_outcome
    ,RIGHT(co.[Month],2) AS outcome_month
    ,LEFT(co.[month],4) AS outcome_year
FROM stage.city_street2 AS cs
LEFT JOIN stage.city_outcome2 AS co
  ON co.[Crime ID] = cs.[Crime_ID]
)
,cte2 AS
(
SELECT
     crime_id
    ,TRY_CAST(CONCAT(crime_year,'-',crime_month,'-','01') AS DATE) AS
     crime_date
    ,TRY_CAST(CONCAT(outcome_year,'-',outcome_month,'-','01') AS DATE) AS
      outcome_date
    ,LSOA_code
    ,crime_type
    ,first_outcome
    ,final_outcome
FROM cte
)
SELECT
     crime_id
    ,CAST(CONCAT(SUBSTRING(CAST(crime_date AS VARCHAR(4)),1,4)
            ,SUBSTRING(CAST(crime_date AS VARCHAR(7)),6,2)) AS INT) AS
     crime_date
    ,CAST(CONCAT(SUBSTRING(CAST(outcome_date AS VARCHAR(4)),1,4)
            ,SUBSTRING(CAST(outcome_date AS VARCHAR(7)),6,2)) AS INT) AS
     outcome_date
    ,CAST(DATEDIFF(mm, crime_date, outcome_date) AS INT) AS
```

```
    months_between
   ,LSOA_code
   ,crime_type
   ,first_outcome
   ,final_outcome
INTO #temptable
FROM cte2
```

Listing 12: Set up City2 Data in temptable

Back to Section (of the above code)

Use temptable to insert City Data into DimCrime

```
INSERT INTO dbo.DimCrime
   (
    crime_id
   ,crime_type_id
   ,first_outcome_id
   ,final_outcome_id
   )
SELECT
    t.crime_id
   ,crime_type_id
   ,ISNULL(first_outcome_id , 0) AS first_outcome_id
   ,ISNULL(final_outcome_id , 0) AS final_outcome_id
FROM #temptable AS t
INNER JOIN DictCrimeType AS ct
  ON ct.crime_type = t.crime_type
LEFT JOIN DictFirstOutcome AS fr
  ON fr.first_outcome = t.first_outcome
LEFT JOIN DictFinalOutcome AS fn
  ON fn.final_outcome = t.final_outcome
```

Listing 13: Use temptable to insert City Data into DimCrime

Back to Section (of the above code)

Use temptable to insert City Data into FactCrime

```sql
INSERT INTO FactCrime
   (
    crime_id
   ,crime_date
   ,LSOA_code
   ,weather_id
   ,pop_id
   )
SELECT
    t.crime_id
   ,t.crime_date
   ,t.LSOA_code
   ,CAST(CONCAT(t.crime_date,b.grid_id) AS BIGINT) AS weather_id
   ,CAST(CONCAT(d.[year],g.borough_id) AS INT) AS pop_id
FROM #temptable AS t
INNER JOIN DimGeog AS g
  ON g.LSOA_code = t.LSOA_code
INNER JOIN DimBorough AS b
  ON b.borough_id = g.borough_id
INNER JOIN DimDate As d
  ON d.date_key = t.crime_date

-- sense check to make sure that this has executed properly - yes,
    32139 rows returned
SELECT
    f.crime_id
FROM FactCrime AS f
INNER JOIN DimCrime AS d
  ON d.crime_id = f.crime_id
```

Listing 14: Use temptable to insert City Data into FactCrime

Back to Section (of the above code)

Clean Met Street Data

```sql
;WITH cte1
AS
(
SELECT
    CASE
     WHEN [Crime ID] IS NULL AND [Crime type] = 'Anti-social behaviour'
       THEN CAST(NEWID() AS VARCHAR(64))
     ELSE [Crime ID]
    END AS crime_id
   ,[Month]
   ,[LSOA code]
   ,[Crime type]
   ,[Last outcome category]
FROM stage.met_street
), cte2
AS
(
SELECT
    *
   ,ROW_NUMBER() OVER (PARTITION BY crime_id ORDER BY crime_id) AS rn
FROM cte1
)
SELECT
    *
INTO stage.met_street2
FROM cte2
--where crime_id IS NULL -- now no nulls!
WHERE 1=1
  AND rn = 1
  AND [LSOA code] IS NOT NULL

-- count = 4,821,478 which means we lost about 200,000 rows!
-- this is fine - a quick eyeball shows they are only double-charges
-- we also don't care if we can't attribute the crime to a borough
-- (discard if there is no LSOA, this is < 1% of data)
```

Listing 15: Clean Met Street Data

Back to Section (of the above code)

Clean Met Outcome Data

```
-- This imore simple, since there are no entries without a crime_id
-- and NULLs are allowed.
-- Create met_outcome2:
;WITH cte
AS
(
SELECT
    [Crime ID]
   ,[Month]
   ,[Outcome type]
   ,ROW_NUMBER() OVER (PARTITION BY [crime id] ORDER BY [crime id]) AS
    rn -- 4176131
FROM stage.met_outcome
)
SELECT
    *
--INTO stage.met_outcome2
FROM cte
WHERE rn = 1 -- complete, 3,117,187 rows ; this is fine,
             -- as a quick eyeball shows only multiple charges.
```

Listing 16: Clean Met Outcome Data

Back to Section (of the above code)
Set up Met2 Data into temptable

```
-- This is similar to the process we went through with the City data
IF OBJECT_ID('tempdb..#temptable') IS NOT NULL
  BEGIN
    DROP TABLE #temptable
  END

;WITH cte
AS
(
SELECT
    ms.[Crime_ID] AS crime_id
   ,RIGHT(ms.[Month],2) AS crime_month
   ,LEFT(ms.[month],4) AS crime_year
   ,ms.[LSOA code] AS LSOA_code
   ,ms.[Crime type] AS crime_type
   ,ms.[Last outcome category] AS first_outcome
   ,mo.[Outcome type] AS final_outcome
   ,RIGHT(mo.[Month],2) AS outcome_month
```

```
    ,LEFT(mo.[month],4) AS outcome_year
FROM stage.met_street2 AS ms
LEFT JOIN stage.met_outcome2 AS mo
  ON mo.[Crime ID] = ms.[Crime_ID]
)
,cte2 AS
(
SELECT
     crime_id
    ,TRY_CAST(CONCAT(crime_year,'-',crime_month,'-','01') AS DATE) AS
     crime_date
    ,TRY_CAST(CONCAT(outcome_year,'-',outcome_month,'-','01') AS DATE) AS
       outcome_date
    ,LSOA_code
    ,crime_type
    ,first_outcome
    ,final_outcome
FROM cte
)
SELECT
     crime_id
    ,CAST(CONCAT(SUBSTRING(CAST(crime_date AS VARCHAR(4)),1,4)
           ,SUBSTRING(CAST(crime_date AS VARCHAR(7)),6,2)) AS INT) AS
     crime_date
    ,CAST(CONCAT(SUBSTRING(CAST(outcome_date AS VARCHAR(4)),1,4)
           ,SUBSTRING(CAST(outcome_date AS VARCHAR(7)),6,2)) AS INT) AS
     outcome_date
    ,CAST(DATEDIFF(mm, crime_date, outcome_date) AS INT) AS
     months_between
    ,LSOA_code
    ,crime_type
    ,first_outcome
    ,final_outcome
INTO #temptable
FROM cte2
-- ~ 4.8 million rows, great!
```

Listing 17: Set up Met2 Data into temptable

Back to Section (of the above code)

Use temptable to insert Met Data into DimCrime

```
INSERT INTO dbo.DimCrime
  (
   crime_id
  ,crime_type_id
  ,first_outcome_id
  ,final_outcome_id
  )
SELECT
   t.crime_id
  ,crime_type_id
  ,ISNULL(first_outcome_id, 0) AS first_outcome_id
  ,ISNULL(final_outcome_id, 0) AS final_outcome_id
FROM #temptable AS t
INNER JOIN DictCrimeType AS ct
  ON ct.crime_type = t.crime_type
LEFT JOIN DictFirstOutcome AS fr
  ON fr.first_outcome = t.first_outcome
LEFT JOIN DictFinalOutcome AS fn
  ON fn.final_outcome = t.final_outcome
-- fine, 4821478 rows!
```

Listing 18: Use temptable to insert Met Data into DimCrime

Back to Section (of the above code)
Use temptable to insert Met Data into FactCrime

```
INSERT INTO FactCrime
  (
   crime_id
  ,crime_date
  ,LSOA_code
  ,weather_id
  ,pop_id
  )
SELECT
   t.crime_id
  ,t.crime_date
  ,t.LSOA_code
  ,CAST(CONCAT(t.crime_date,b.grid_id) AS BIGINT) AS weather_id
  ,CAST(CONCAT(d.[year],g.borough_id) AS INT) AS pop_id
FROM #temptable AS t
INNER JOIN DimGeog AS g
  ON g.LSOA_code = t.LSOA_code
INNER JOIN DimBorough AS b
```

```
  ON b.borough_id = g.borough_id
INNER JOIN DimDate As d
  ON d.date_key = t.crime_date
-- DONE, 4,816,276 rows!!

-- NB: FactCrime only accepts LSOA codes from  Greater London District
--     Hence why there are more rows in FactCrime than DimCrime,
--     the latter being independent of geog

-- We can verify this by simply checking some of these LSOA codes:

SELECT
    lsoa_code
FROM #temptable
WHERE lsoa_code NOT IN
            (
            SELECT
                lsoa_code
            FROM DimGeog
            )
-- checking these on a map verify that they are indeed outside
-- of the Greater London boundary
```

Listing 19: Use temptable to insert Met Data into FactCrime

Back to Section (of the above code)
Create Outcome Type Temporary Dictionaries

```
-- we will use some #temptables in order to create temporary
    dictionaries for the category of outcome

IF OBJECT_ID('tempdb..#tempfirst') IS NOT NULL
  BEGIN
    DROP TABLE #tempfirst
    DROP TABLE #tempfinal
  END

-- Frst Outcome Dictionary:
;WITH ctefirst
AS
(
SELECT
    *
   ,CASE
    WHEN first_outcome_id IN (0, 1, 4, 8, 11, 15, 16, 18, 20, 22, 23,
    24, 25, 26) THEN 1
    WHEN first_outcome_id IN (12, 17, 19) THEN 3
```

```
    ELSE 2
  END AS first_cat_id
FROM DictFirstOutcome
)
SELECT
  *
  ,CASE
    WHEN first_cat_id = 1 THEN 'No Consequence'
    WHEN first_cat_id = 2 THEN 'Penalty Issued'
    ELSE 'Warning'
  END AS first_cat
INTO #tempfirst
FROM ctefirst

-- Final Outcome Dictionary:
;WITH ctefinal
AS
(
SELECT
  *
  ,CASE
    WHEN final_outcome_id IN (0, 1, 9, 13, 15, 17, 20, 21, 22, 23) THEN
     1
    WHEN final_outcome_id IN (10, 14, 16) THEN 3
    ELSE 2
  END AS final_cat_id
FROM DictFinalOutcome
)
SELECT
  *
  ,CASE
    WHEN final_cat_id = 1 THEN 'No Consequence'
    WHEN final_cat_id = 2 THEN 'Penalty Issued'
    ELSE 'Warning'
  END AS final_cat
INTO #tempfinal
FROM ctefinal
```

Listing 20: Create Outcome Type Temporary Dictionaries

Back to Section (of the above code)

Evaluate Warning and Outcome Ratios; Insert into Export Table

```
-- Now evaluate the warning and penalty ratios and throw it into tableau.police_efficiency
;WITH ctecount
AS
(
SELECT DISTINCT
    b.borough_name
   ,d.[year]
   ,d.[month]
   ,f.weather_id
   ,p.[population]

   ,COUNT(CASE WHEN t1.first_cat_id = 3 THEN 1 ELSE NULL END)
     OVER(PARTITION BY b.borough_id, d.[year], d.[month] ORDER BY d.[year]) AS warning_count1
   ,COUNT(CASE WHEN t1.first_cat_id = 1 AND t2.final_cat_id = 3 THEN 1 ELSE NULL END)
     OVER(PARTITION BY b.borough_id, d.[year], d.[month] ORDER BY d.[year]) AS warning_count2

   ,COUNT(f.crime_id)
     OVER(PARTITION BY b.borough_id, d.[year], d.[month] ORDER BY d.[year]) AS crime_count

   ,COUNT(CASE WHEN t1.first_cat_id = 2 THEN 1 ELSE NULL END)
     OVER(PARTITION BY b.borough_id, d.[year], d.[month] ORDER BY d.[year]) AS penalty_count1
   ,COUNT(CASE WHEN t1.first_cat_id = 1 AND t2.final_cat_id = 2 THEN 1 ELSE NULL END)
     OVER(PARTITION BY b.borough_id, d.[year], d.[month] ORDER BY d.[year]) AS penalty_count2

FROM FactCrime AS f
INNER JOIN DimCrime AS c
  ON c.crime_id = f.crime_id
INNER JOIN DimDate AS d
  ON d.date_key = f.crime_date
INNER JOIN DimGeog AS g
  ON g.LSOA_code = f.LSOA_code
INNER JOIN DimBorough AS b
  ON b.borough_id = g.borough_id
INNER JOIN DimPop AS p
  ON p.pop_id = f.pop_id
INNER JOIN #tempfirst AS t1
  ON t1.first_outcome_id = c.first_outcome_id
INNER JOIN #tempfinal AS t2
```

```
  ON t2.final_outcome_id = c.final_outcome_id
),
cteratio AS
(
SELECT
    c.borough_name
   ,c.[year]
   ,c.[month]
   ,warning_count1 + warning_count2 AS warning_count
   ,crime_count
   ,[population]
   ,penalty_count1 + penalty_count2 AS penalty_count
   ,w.temp
   ,w.rainfall
   ,w.sun_hours
FROM ctecount AS c
INNER JOIN DimWeather AS w
  ON w.weather_id = c.weather_id
)
SELECT
    borough_name AS Borough
   ,[Year]
   ,[Month]
   ,CAST(warning_count AS FLOAT) / CAST(crime_count AS FLOAT) AS [
    Warning Ratio]
   ,CAST(penalty_count AS FLOAT) / CAST(crime_count AS FLOAT) AS [
    Penalty Ratio]
   ,(CAST(crime_count AS FLOAT) / CAST([population] AS FLOAT))*1000 AS [
    Crime Count per 1000pop]
   ,temp AS [Temp (C)]
   ,rainfall AS [Rainfall (mm)]
   ,sun_hours AS [Sunlight Hours]
INTO tableau.police_efficiency
FROM cteratio
GO

-- sense check rowcount returns the correct 1980 rows for 33 boroughs,
    12 months, 5 years!
```

Listing 21: Evaluate Warning and Outcome Ratios; Insert into Export Table

Back to Section (of the above code)

Create Crime Type Dictionary

```sql
;WITH ctetype
AS
(
SELECT
    *
   ,CASE
     WHEN crime_type_id IN (1, 2, 11) THEN 3
     WHEN crime_type_id IN (3, 4, 6, 13, 14) THEN 2
     WHEN crime_type_id = 16 THEN 4
     ELSE 1
   END AS crime_cat_id
FROM DictCrimeType
)
SELECT
    *
   ,CASE
     WHEN crime_cat_id = 1 THEN 'Theft'
     WHEN crime_cat_id = 2 THEN 'Violent'
     WHEN crime_cat_id = 3 THEN 'Antisocial'
     ELSE 'Other'
    END AS crime_cat
INTO #temptype
FROM ctetype
```

Listing 22: Create Crime Type Dictionary

Back to Section (of the above code)
Create Weather Trend Export Table

```sql
;WITH ctecrime
AS
(
SELECT DISTINCT
    borough_name
   ,d.[year]
   ,[month]
   ,f.weather_id
   ,p.[population]

   ,COUNT(CASE WHEN t.crime_cat_id = 1 THEN 1 ELSE NULL END)
     OVER(PARTITION BY b.borough_id, d.[year], d.[month]) AS theft_count
   ,COUNT(CASE WHEN t.crime_cat_id = 2 THEN 1 ELSE NULL END)
     OVER(PARTITION BY b.borough_id, d.[year], d.[month]) AS
     violent_count
```

```sql
  ,COUNT(CASE WHEN t.crime_cat_id = 3 THEN 1 ELSE NULL END)
    OVER(PARTITION BY b.borough_id, d.[year], d.[month]) AS antisocial_count
  ,COUNT(CASE WHEN t.crime_cat_id = 4 THEN 1 ELSE NULL END)
    OVER(PARTITION BY b.borough_id, d.[year], d.[month]) AS other_count

FROM FactCrime AS f
INNER JOIN DimCrime AS c
  ON c.crime_id = f.crime_id
INNER JOIN DimDate AS d
  ON d.date_key = f.crime_date
INNER JOIN DimGeog AS g
  ON g.LSOA_code = f.LSOA_code
INNER JOIN DimBorough AS b
  ON b.borough_id = g.borough_id
INNER JOIN DimPop AS p
  ON p.pop_id = f.pop_id
INNER JOIN #temptype AS t
  ON t.crime_type_id = c.crime_type_id
)
SELECT
   borough_name
  ,[year]
  ,[month]
  ,(CAST(theft_count AS FLOAT) / CAST([population] AS FLOAT))*1000 AS [Theft Count per 1000pop]
  ,(CAST(violent_count AS FLOAT) / CAST([population] AS FLOAT))*1000 AS [Violent Count per 1000pop]
  ,(CAST(antisocial_count AS FLOAT) / CAST([population] AS FLOAT))*1000 AS [Anti-social Count per 1000pop]
  ,(CAST(other_count AS FLOAT) / CAST([population] AS FLOAT))*1000 AS [Other Count per 1000pop]
  ,temp
  ,rainfall
  ,sun_hours
INTO tableau.weather_trend
FROM ctecrime As c
INNER JOIN DimWeather AS w
  ON w.weather_id = c.weather_id

-- complete, 1980 rows as it should be.
```

Listing 23: Create Weather Trend Export Table

Back to Section (of the above code)

Create Unpivoted Weather Trend View

```
-- create the unpivoted view so that we can aggregate by crime type.
-- OLD ONE only gives a general idea of the correlation
CREATE VIEW weather_trend
AS

SELECT
    borough_name
   ,[year]
   ,[month]
   ,[count]
   ,cat
   ,temp
   ,rainfall
   ,sun_hours
FROM
(
SELECT
    borough_name
   ,[year]
   ,[month]
   ,[theft count per 1000pop]
   ,[violent count per 1000pop]
   ,[anti-social count per 1000pop]
   ,[other count per 1000pop]
   ,temp
   ,rainfall
   ,sun_hours
FROM tableau.weather_trend
) AS unp
UNPIVOT
(
   [count]
  FOR [cat]
    IN  (
         [theft count per 1000pop]
        ,[violent count per 1000pop]
        ,[anti-social count per 1000pop]
        ,[other count per 1000pop]
        )
) AS upp
```

Listing 24: Create Unpivoted Weather Trend View

Back to Section (of the above code)

Create View : Seasonal Variance by Crime Type

```
-- First look at Anti-Social Behaviour Crimes.
-- The other Categories {Violent, Theft, Other} are almost identical
-- Only change the name of the crime type.

CREATE VIEW seaonal_var_ASB
AS

WITH cte
AS
(
SELECT DISTINCT
    borough_name
    ,[month]
    ,AVG([Anti-social Count per 1000pop]) OVER (PARTITION BY [month]) AS
     ASB
    ,AVG(rainfall) OVER (PARTITION BY [month]) AS rain
    ,AVG(temp) OVER (PARTITION BY [month]) AS temp
    ,AVG(sun_hours) OVER (PARTITION BY [month]) AS sun
FROM tableau.weather_trend
),
cte2
AS
(
SELECT
    borough_name
    ,[month]
    ,ASB
    ,LAG(ASB,1,0) OVER (PARTITION BY borough_name ORDER BY [month]) AS
     prev_ASB
    ,rain
    ,LAG(rain,1,0) OVER (PARTITION BY borough_name ORDER BY [month]) AS
     prev_rain
    ,temp
    ,LAG(temp,1,0) OVER (PARTITION BY borough_name ORDER BY [month]) AS
     prev_temp
    ,sun
    ,LAG(sun,1,0) OVER (PARTITION BY borough_name ORDER BY [month]) AS
     prev_sun
FROM cte
)
,
cte3
AS
(
SELECT
    [month]
```

```
    ,ASB
    ,CASE
      WHEN prev_ASB = 0 THEN (3.19514114905938)
      ELSE prev_ASB
     END AS prev_ASB
    ,rain
    ,CASE
      WHEN prev_rain = 0 THEN (62.4495757575758)
      ELSE prev_rain
     END AS prev_rain
    ,temp
    ,CASE
      WHEN prev_temp = 0 THEN (10.3188484848485)
      ELSE prev_temp
     END AS prev_temp
    ,sun
    ,CASE
      WHEN prev_sun= 0 THEN (52.8869090909091)
      ELSE prev_sun
     END AS prev_sun
FROM cte2
)
SELECT DISTINCT
     [month]
    ,(ASB - prev_ASB) / prev_ASB AS [ASB % Diff]
    ,(rain - prev_rain) / prev_rain AS [rain % Diff]
    ,(temp - prev_temp) / prev_temp AS [temp % Diff]
    ,(sun - prev_sun) / prev_sun AS [sun % Diff]
FROM cte3
GO
```

Listing 25: Create Crime View : Seasonal Variance by Crime Type

Back to Section (of the above code)

Create Forecast Export Table

```
-- First create an export table with the appropriate information.
-- This example uses 2016 June Data to forecast July Data
;WITH cte
AS
(
SELECT
    [borough_name]
   ,[year]
   ,[month]
   ,[Theft Count per 1000pop] AS last_theft
   ,[Violent Count per 1000pop] AS last_violent
   ,[Anti-social Count per 1000pop] AS last_ASB
   ,[Other Count per 1000pop] AS last_other
   ,[temp] AS last_temp
      ,[rainfall] As last_rain
      ,[sun_hours] AS last_sun
FROM tableau.weather_trend
WHERE 1=1
  AND [year] = 2016
  AND [month] = 6
),
cte2
AS
(
SELECT
    c.*
   ,[temp] AS forecast_temp
      ,[rainfall] As forecast_rain
      ,[sun_hours] AS forecast_sun
FROM cte AS c
INNER JOIN tableau.weather_trend AS w
  ON w.borough_name = c.borough_name
  AND w.[year] = 2016
  AND w.[month] = 7
),
cte3
AS
(
SELECT
    borough_name
   ,[year]
   ,[month]
   ,last_theft
   ,last_violent
   ,last_ASB
   ,last_other
```

```
    ,(forecast_temp - last_temp) / last_temp AS [temp%]
    ,(forecast_rain - last_rain) / last_rain AS [rain%]
    ,(forecast_sun - last_sun) / last_sun AS [sun%]
FROM cte2
),
cte4
As
(
SELECT --* from cte3
    borough_name
    ,[year]
    ,[month]
    ,last_theft
    ,last_violent
    ,last_ASB
    ,last_other
    ,1 + 0.7530688*([temp%]+[rain%]+[sun%])/3 AS [crime change coeff] --
    MODIFY % COEFFICIENT HERE
FROM cte3
)
SELECT
    borough_name
    ,[year]
    ,[month]
    ,last_theft
    ,last_violent
    ,last_ASB
    ,last_other
    ,last_theft * [crime change coeff] AS new_theft
    ,last_violent * [crime change coeff] AS new_violent
    ,last_ASB * [crime change coeff] AS new_ASB
    ,last_other * [crime change coeff] As new_other
INTO tableau.weather_forecast
FROM cte4
```

Listing 26: Create Crime Forecast Export Table

Back to Section (of the above code)

Intermediate Crime Count Forecast

```
-- Make a 'Count Version' (with absolute numbers, not %)
;WITH cte
AS
(
SELECT
    w.borough_name
    ,p.[population]
```

```
      ,[last_theft]
      ,[last_violent]
      ,[last_ASB]
      ,[last_other]
      ,[new_theft]
      ,[new_violent]
      ,[new_ASB]
      ,[new_other]
FROM tableau.weather_forecast AS w
INNER JOIN DimBorough AS b
  ON b.borough_name = w.borough_name
INNER JOIN DimPop As p
  ON p.borough_id = b.borough_id
  AND p.[year] = w.[year]
),
cte2
AS
(
SELECT
    borough_name
   ,[last_theft]*[population]/1000  AS last_theft_count
      ,[last_violent]*[population]/1000  AS last_violent_count
      ,[last_ASB]*[population]/1000  AS last_ASB_count
      ,[last_other]*[population]/1000  AS last_other_count
      ,[new_theft]*[population]/1000  AS new_theft_count
      ,[new_violent]*[population]/1000  AS new_violent_count
      ,[new_ASB]*[population]/1000  AS new_ASB_count
      ,[new_other]*[population]/1000  AS new_other_count

FROM cte
)
SELECT
    borough_name
   ,new_theft_count - last_theft_count AS theft_change
   ,new_violent_count - last_violent_count AS violent_change
   ,new_ASB_count - last_ASB_count AS ASB_change
   ,new_other_count - last_other_count As last_change
INTO tableau.crime_count_forecast
FROM cte2
```

Listing 27: Intermediate Crime Count Forecast

Back to Section (of the above code)

Create View : Crime Change Forecast

```
-- Finally, export as view:
CREATE VIEW crime_change_forecast
AS

SELECT
    borough_name
   ,[change count]
   ,[change type]
FROM
(
SELECT [borough_name]
        ,[theft_change]
        ,[violent_change]
        ,[ASB_change]
        ,[last_change]
FROM [tableau].[crime_count_forecast]
) AS unp
UNPIVOT
(
   [change count]
   FOR [change type] IN
     (
       [theft_change]
      ,[violent_change]
      ,[ASB_change]
      ,[last_change]
     )
) AS upp
```

Listing 28: Create View : Crime Change Forecast

Back to Section (of the above code)

Test Forecast Model

```

-- Dictionary first, makes it easier for us.
IF OBJECT_ID('tempdb..#tempttype') IS NOT NULL
   BEGIN
      DROP TABLE #temptable
      DROP TABLE #tempjuly
      DROP TABLE #temppivotjuly
      DROP TABLE #tempjune
```

```
    DROP TABLE #temppivotjune
    DROP TABLE #tempforecast
    DROP TABLE #temptemp
  END

;WITH ctetype
AS
(
SELECT
   *
  ,CASE
    WHEN crime_type_id IN (1, 2, 11) THEN 3
    WHEN crime_type_id IN (3, 4, 6, 13, 14) THEN 2
    WHEN crime_type_id = 16 THEN 4
    ELSE 1
  END AS crime_cat_id
FROM DictCrimeType
)
SELECT
   *
  ,CASE
    WHEN crime_cat_id = 1 THEN 'Theft'
    WHEN crime_cat_id = 2 THEN 'Violent'
    WHEN crime_cat_id = 3 THEN 'Antisocial'
    ELSE 'Other'
   END AS crime_cat
INTO #temptype
FROM ctetype

-- test forecast model = 2016-07
SELECT DISTINCT
   b.borough_name
  ,tt.crime_cat
  ,COUNT(f.crime_id) OVER (PARTITION BY b.borough_id, tt.crime_cat  )
   AS july_count
INTO #tempjuly
FROM FactCrime AS f
INNER JOIN DimGeog AS g
  ON g.LSOA_code = f.LSOA_code
INNER JOIN DimBorough AS b
  ON b.borough_id = g.borough_id
INNER JOIN DimCrime AS c
  ON c.crime_id = f.crime_id
INNER JOIN #temptype AS tt
  ON tt.crime_type_id = c.crime_type_id
WHERE f.crime_date = 201607
GO

-- great, now we have to pivot this data...
```

```sql
SELECT
    borough_name
   ,theft
   ,antisocial
   ,violent
   ,other
INTO #temppivotjuly
FROM
(
SELECT
    borough_name
   ,crime_cat
   ,july_count
FROM #tempjuly
) AS sourcetable
PIVOT
(
  AVG(july_count)
  FOR crime_cat IN
     (
      theft
     ,antisocial
     ,violent
     ,other
     )
) AS pvv

-- now we want to know what we forecasted for july, and compare them.
SELECT DISTINCT
    b.borough_name
   ,tt.crime_cat
   ,COUNT(f.crime_id) OVER (PARTITION BY b.borough_id, tt.crime_cat )
    AS june_count
INTO  #tempjune
FROM FactCrime AS f
INNER JOIN DimGeog AS g
  ON g.LSOA_code = f.LSOA_code
INNER JOIN DimBorough AS b
  ON b.borough_id = g.borough_id
INNER JOIN DimCrime AS c
  ON c.crime_id = f.crime_id
INNER JOIN #temptype AS tt
  ON tt.crime_type_id = c.crime_type_id
WHERE f.crime_date = 201606

-- and unpivot
SELECT
    borough_name
   ,theft
```

```sql
    ,antisocial
    ,violent
    ,other
INTO #temppivotjune
FROM
(
SELECT
     borough_name
    ,crime_cat
    ,june_count
FROM #tempjune
) AS sourcetable
PIVOT
(
  AVG(june_count)
  FOR crime_cat IN
      (
       theft
      ,antisocial
      ,violent
      ,other
      )
) AS pvv

-- Now get the forecast values for july
;WITH ctejune
AS
(
SELECT
     j.borough_name
    ,theft
    ,antisocial
    ,violent
    ,other
    ,theft_change
    ,violent_change
    ,ASB_change
    ,last_change
FROM #temppivotjune As j
INNER JOIN tableau.crime_count_forecast3 AS fc
  ON fc.borough_name = j.borough_name
)
SELECT
     borough_name
    ,theft + theft_change AS theft_july_forecast
    ,antisocial + ASB_change AS antisocial_july_forecast
    ,violent + violent_change AS violent_july_forecast
    ,other + last_change AS other_july_forecast
INTO #tempforecast
```

```
FROM ctejune
GO

-- Penultimately, get the percentage difference between
-- predicted and actual..
;WITH ctepercent
AS
(
SELECT
    t.borough_name
   ,theft
   ,antisocial
   ,violent
   ,other
   ,antisocial_july_forecast
   ,theft_july_forecast
   ,violent_july_forecast
   ,other_july_forecast
FROM #temppivotjuly AS t
INNER JOIN #tempforecast As f
  ON f.borough_name = t.borough_name
)
SELECT
    borough_name
   ,100*(theft_july_forecast - theft) / theft AS theft_p_diff
   ,100*(antisocial_july_forecast - antisocial) / antisocial AS
    antisocial_p_diff
   ,100*(violent_july_forecast - violent) / violent AS violent_p_diff
   ,100*(other_july_forecast - other) / other AS other_p_diff
INTO #temptemp
FROM ctepercent


-- Finally, the overall inaccuracy for each borough:
SELECT
    borough_name
   ,(theft_p_diff + antisocial_p_diff + violent_p_diff + theft_p_diff) /
     4 AS [% Inaccuracy]
FROM #temptemp

-- AVG of this = -5.52%
-- The forecast model underestimates the data by 5.52%
```

Listing 29: Test Forecast Model

Back to Section (of the above code)