TASK 5

II. To improve the quality of the data file, we should seek to fill in the missing   
values of column ‘Gap (s)’. For simplicity, let us focus on the NB\_MID lane   
(North direction) only, while considering any Tuesday between 7:00 and   
18:59:59 o’clock for which the value for this column is missing.   
   
But, before choosing a method to fill in the missing values of column ‘Gap   
(s)’, using Python, build a profile of the column’s data values to find out its   
relevant details. This profile of the values present in the column should help   
you decide how the missing values should be filled, by providing you with an   
idea of what data values could possibly represent as best as possible the   
“real” values that are missing, given the present data and use case at hand.   
   
Following the profiling exercise, enumerate the profiling techniques you used   
and what each one has allowed you to find out about the data, using the   
relevant text field (the one associated with this task) in the relevant   
Blackboard Test (the one associated with this coursework).   
   
Also, following the instructions provided in Section 2, upload to Gitlab any   
commented/explained Python code and any associated plotted graphs you

6   
generated for this task, describing method application and your decision-  
making.   
(HINT: try to obtain a detailed profile of the column (rather than a minimal one).

1) Descriptive Statistics:

The resultant DDS values, found in the code, provide us with simple characteristics of the data analyzed, and can help in first-order analysis that guide further experiments and profiling. From the values obtained, we immediately see that mean is greater than median, which suggests right-skewed data. This is further solidified through a high skewness value, which means there are a few large values far away from the median. The kurtosis value implies a heavy leptokurtic distribution, which means that there is a sharp peak and a heavy tail, indicating extreme values (outliers), compared to a normal distribution.

When we group these findings together, we can conclude that although most of the Tuesday gap values fall within a somewhat predictable range, there is a lot of outliers that cannot be ignored. This might also suggest that during the inspected period, there are times where gaps are relatively small, and times where the gaps visibly increase.

2) Completeness of NB\_MID:

Here, we have checked whether the completeness of the NB\_MID lane like that of the whole dataframe, from previous task. It was found that this value is around 98.03%, which indicates the data is highly complete. However, it would be beneficial to further investigate why the 1.97% of data is missing, and when that data is missing, as it might introduce some bias to analysis (for example, most missing values connected to a single period). However, if these values seem random, we could simply ignore them, especially with high completeness.

3) Distribution Visualization through a histogram:

To generate a more insightful view and context of the previously made points, we have generated a histogram of NB\_MID gap values, as seen in Figure 1 in the attached pdf. We notice high right-skewness, sharp peak at the beginning, and presence of long tail with extreme outliers, which corresponds to the points found in the DDS section. Again, we see that most of the time the gaps between cars are relatively short, which could indicate a pattern or regularity in the data. However, while most gaps are short, there might be prolonged periods when these values increase, during off-peak hours for example. As we do not know the source of this, we should investigate further and check other trends in data to evaluate the outliers’ legitimacy.

4) DDS Visualization through boxplots:

To check whether trends are consistent across the month, a set of weekly boxplots has been generated, as seen in Figure 2 of the PDF. First, we can see is the consistency of IQRs, medians, and high outliers across all weeks, suggesting that the gap behavior can be generalized across the month without introducing much error or bias. Although one can see some variations on a week-to-week basis (for example, in outliers), these are relatively small, and can therefore be ignored. Again, we see similar behaviors noticed before, like high clustering around a relatively low median, and presence of high outliers across all plots. These plots indicate that a week-based analysis would result in virtually the same results as a more general four-week analysis. Therefore, we can justifiably analyze whole data, instead of on a weekly basis.

5) Scatter Plot Visualization through Gap as a function of time:

Here, to check whether the gap values change across different times of day, we have generated the resultant scatter plot, as seen in Figure 3 of the PDF. We can see that across all hours, data points are clustered towards the bottom. However, there are noticeable variations across the day. Through: morning hours (7-9am), the gaps are tightly clustered, with less extreme outliers; midday (10-12pm), there is a slight drop in gap distribution, with more extensive outliers; afternoon (1-3pm), we see that the gap distribution and outliers’ presence increase; evening (4-6pm), the gaps cluster together, and outliers’ severity drops. We can conclude that during peak hours (morning rush and evening way home), the gaps remain consistent and more predictable, whereas during off-peak hours they become much more varied and harder to predict, due to much more relaxed traffic flow.

6) Correlation Factor:

Here, we have generated Pearson correlation factors to check the gap dependency on other attributes. We see the following: extremely weak correlation between gap and day hour; weak correlation between gap and speed; very strong correlation between gap and headway. We see that the headway attribute can prove extremely helpful when estimating missing gap data, and vice versa. This can be further supported through the speed attribute, but very carefully, due to a somewhat weak correlation. As expected, time of day is not a good estimator, as the gap value distribution is only somewhat affected by the hour attribute. Thanks to these, we can more adequately fill in data, without having to worry about introducing too much error/bias.

III. Considering the variety of ways in which a numeric column, such as ‘Gap   
(s)’, can have its empty cells filled with values, choose a method to fill in the   
missing values for this particular column (from the suggested below), having   
as a basis the profile you generated in Task 5.II, and considering the use   
case at hand. For facilitating the marking of this task, choose one of the   
following methods:   
1) replacing all missing values with a single constant; or   
2) replacing all missing values with the AVG of all the column values; or   
3) replacing each missing value with the AVG of the column values of a   
given group of records; or   
4) replacing all missing values with the median value of the column;   
   
Write in the relevant answer text field (the one associated with this task) of   
the relevant Blackboard Test (the one associated with this coursework) the   
method you chose (from the list provided above) to fill in the missing values   
of column ‘Gap (s)’, making sure your choice of method is clear, by   
explicitly specifying what it is, and making sure your choice is justified.   
   
Additionally, suggest one method, outside of the above list, which is (also?   
or more?) suitable according to the profile of this particular column. Justify   
your choice using the profile you generated in Task 5.II as a basis, and   
considering the use case at hand. Write it all on the relevant Blackboard   
Test.   
   
Following the instructions provided in Section 2, upload to Gitlab any   
commented/explained Python code and associated plotted graphs you   
generated for this task, describing method application and your decision-  
making.  
  
Method chosen from the given options: 3) replacing each missing value with the AVG of the column values of a given group of records.

The first thing we noticed is the extremely high right-skewness of the gap data (Figure 1). This indicates that, although most gaps are short, there are longer gaps that cannot be excluded. If we were to apply a constant, or a median, we would not be representing the data appropriately, as we would be simplifying the trends too much. A simple avg is also not a good idea, due to high frequency of outliers, which would greatly affect its value.

Moreover, we have also noticed that there is, although not that easy to read, time dependency of gaps during the day (Figure 3). We see that the gaps are much more predictable during peak hours (morning and evening commute). On the other hand, during off-hours we see the behavior become less predictable and varied. This means that using a single value, without caring for trends, for all missing values would decrease the accuracy of our findings.

Additionally, we have noticed a strong correlation with the headway attribute. As both attributes capture the time distance of two vehicles, we can use one to predict the other accurately.

Given these findings, we see that method chosen makes the most sense in this context. Through applying an average gap value for groups of records of other attributes, such as similar headway or particular hour values, we would be more accurately capturing the patterns and nuances in our data. We would also ensure a genuine representation of the traffic behavior for these missing values, especially during less predictable periods. This would in turn result in more accurate and trust-worthy findings, better than those that a median/avg/constant could provide.

Additional method chosen outside the list: using the most probable value (model-based regression).  
  
As we have previously established, there exists an evident correlation between gap and headway attributes. We can successfully utilize this finding and teach the regression model on the non-missing gap data using headway as the independent variable. Thanks to this, we could achieve accurate predictions, which could be further improved with introduction of more data or attributes. This is extremely important for dynamic data, such as the traffic dataset, as it ensures the method viability through different periods and trends.

Moreover, real-life variables are often multi-dimensional and dependent on a lot of factors, which simple filling methods are more prone to miss, and therefore produce less accurate results. Model-based methods, such as regression, are capable of learning these nuances much better, especially when more variables are introduced. Thanks to this, we could try and utilize different combinations of variables (headway, speed, hour), and see which ones reflect the traffic nature best.

As traffic data is prone to random behavior due to accidents, roadworks, social events, and even traffic on other parts of the road, it is crucial to reflect its behavior accurately. A more robust method, such as regression, not only captures the general trend of the gap attribute, but it also considers its surrounding relevant attributes.

Of course, if we rely too much on few variables, we are very prone to overfit. Also, if the change in data/trend is too sudden or sharp, our model might not capture this and treat it as noise.

Summary:

Although the portion of missing data is very small, it is still important to ensure thorough accuracy at all stages and features of the dataset. One could argue that simply ignoring missing data would not be harmful, but we should try avoiding whenever possible, as we might miss out on some important findings. Instead, we should focus on filling the data in a way that represent the real-world as much as possible.

Both methods chosen, averaging with grouping and regression, aim to achieve just that. Of course, the first being much simpler, is much faster and easier to apply, but might fail to consider more complex relations withing the data. On the other hand, regression is usually much more accurate, but can also fail spectacularly if not tuned correctly or developed on wrong underlying assumptions (such as misguided variables).

However, both these methods capture the behavior of traffic data much better than a simple filling with a constant/median/avg of the column. In this context, data integrity is key, as inaccurate findings might result in hurtful decision regarding urban planning, such as costly and inefficient design decisions.

TASK 6

I. To further polish your analytical skills, using Python, estimate the typical   
Friday Journey Time (JT) for the road fragment between site 1083 and site   
1415, between 17:00 and 17:59:59, using only the North direction lanes. JT   
is a very popular traffic application among road users, via which the length of   
time period that it takes to travel through a road fragment is estimated.   
   
You can use the very simple formula for estimating JT, described as follows:   
   
JT = distance/speed   
   
Assume that the distance between the two points in these two sites is   
4.86km. Note the units of distance and time and consider when conversion   
between units is necessary. Also, use the average speed, considering the   
relevant time period (17:00 and 17:59:59) and lanes. To calculate the   
average speed between 17:00 and 17:59:59 of all North lanes, consider not   
only the three North lanes of site 1083, but also the two North lanes of site   
1415. Make sure your JT is given in minutes, which is what road users   
normally get from popular applications such as Google Maps.   
   
Write as, an answer to this question, any data quality checks/profiling

7   
and/or data preparation you had to do prior to calculating JT and any   
challenges you encountered when doing the calculation.   
   
Following the instructions provided in Section 2, upload your commented   
Python code and any plotted graphs to Gitlab.

After filtering all data according to the task at hand through simple functions, we have checked the completeness of the speed attribute for both sites. We have found the completeness to be at 100% for both, meaning our data is full and therefore will not require any filling.

To get the initial idea about our data, we then calculated simple DDS measured through pandas .describe() function. We found that both sets share extremely similar values, meaning both sites represent virtually the same traffic flow and trend, which meant that no standardization/feature extraction was necessary to reduce bias. We have also noticed that mean and median are very close, which pointed towards a symmetrical distribution with little skewness. However, max values for both sites were very far from central tendencies, which prompted further investigation.

As seen in Figures 4 and 5, we notice outliers on both sides of the range. However, the upper parts seem much more pronounced, pulling the whole plots up. It was then decided that additional profile of the data would be acquired. This was done in the form of histograms, as seen in Figures 6 and 7. We notice, as predicted, very symmetrical plots, with most of the outliers landing above the 50mph mark. If we were to ignore these outliers, we would achieve smaller skewness of the distribution, which would be a more representative of the actual traffic trends in that area. Furthermore, these values would be considered extreme in the context of speed limits and regulations, which further suggests that these values can be treated as noise. Therefore, instead of the usual Q3 + 1.5IQR rule for outliers, visible in previous boxplots, it was decided that all speed values above (and including) 50 would be removed from both sets.

Then, these sets were combined into one, and a weighed mean was calculated. This was done because both sites share similar trends, and there was a visible discrepancy in sizes of relevant speed sets. As this value was in the [miles/h] unit, we had to convert it to [km/h] by multiplying it by a constant. Then, we applied the given equation and have achieved the required JT in [hours], which was then converted into [minutes], as specified by the task.

The biggest challenge when tackling this problem was deciding when to draw the line regarding the outliers. If we were to follow the boxplot suggestions, we might have skipped some genuine values. However, the cut-off point of 50mph was chosen rather visually without much further context about the sites at hand, or accurate and exhaustive information about the sensors used. Given more context, the point might have been chosen differently.

Second problem was of course appropriately estimating the JT. This required checking whether these two sites share similar trends, how the datasets sizes differ from each other, and how to properly balance these issues to get an accurate result. Thankfully, the trends were similar, and the “expected” speed value was simply calculated by a weighed mean. The rest of the task was rather straightforward.

TASK 7

I. Choose a technology other than Python (from the suggested in Section 4)   
and repeat Task\_6.I.   
   
Make sure you paste/export the step-by-step code/recipe you developed   
using the technology of your choice. As detailed in Section 2, upload any   
code or exported recipe representation to Gitlab, giving the relevant file the   
name ‘SolutionTask\_7-I’.   
   
Also, describe the features of the technology, emphasising the ones that   
have made your work easier or more difficult, in contrast to Python. In your   
description, you can make comments about functionality that is not offered   
within the technology, rendering you unable to satisfactorily complete Task   
6.I. Write this in the relevant answer text field of the relevant Blackboard   
Test, as described in Section 2

Similarly to previous coursework, I have decided to opt for the MySQL option as an alternative, due to previous experience and relative ease of transferring logic between Python’s pandas and SQL queries.

Due to SQL’s core design for data manipulation and retrieval, its application allowed for more direct, concise, and precise queries, even when considering multiple tables in one go. This is especially useful and easy to use with the added continuous database structure visibility when working with applicable GUIs, such as MySQL Workbench, which can be somewhat achieved in Python’s Pandas describe, but it’s much worse in terms of visibility. Moreover, MySQL inherently supports a lot of helpful functions when dealing with different types of data, which inherently ensure proper data integrity and manipulation. This made the work a bit more structured and condensed.

Of course, as MySQL was designed to handle data exclusively, it is faster when it comes to performance, which is especially visible when dealing with larger datasets. This was more visible this time when compared to previous coursework, as two datasets had to be queried instead of one. However, this was only slightly helpful and reduced the runtime by a margin.

One of the biggest advantages of MySQL over Python is that it was designed to handle databases, which therefore inherently ensures more security over aspects such as data integrity, type management, and constraints, resulting in a very controlled environment, where the user can have more trust in what is happening “under the hood”. It also allows for more clarity and readability, which is especially useful with complex queries. Moreover, aggregating functions or creating temporary tables is much easier and clearer than in Python, which further helps in understanding the operations at hand. This verbosity of course helped tremendously when debugging. However, this required some level of previous experience with the logic behind.

Unfortunately, using MySQL requires much more setup overhead, which can lead to lost time spent on trying to get the server started properly. Again, there were issues regarding file permissions, even when running as a root, which resulted in tedious debugging. Moreover, running queries required having a constant and stable connection to the server. If it becomes unstable or crashes, the unsaved work will be lost with no way of getting it back, which made the work challenging, as it has happened to me twice due to problems with configurations on MacOS. Furthermore, as SQL’s logic of handling queries differs a bit from the ”top-to-bottom” logic of Python (and other languages), this made writing code and debugging a bit tricky. This was evident especially when nesting or aggregating.

Another problem was lack of general utility and flexibility. When trying to do anything outside the MySQL box, such action would be met with an error or rejection. Any form of plotting, even of the simplest plots, is not supported. This can be somehow mitigated through “tricks” or external software but requires additional work. As data profiling and general cleaning was done before using MySQL, this was not that big of a problem. However, if these parts were supposed to be done exclusively in MySQL, they would become very difficult and awkward to work with. Moreover, it would be immeasurably harder to notice trends, patterns, or special points of interests. Simply outputting numbers and trying to figure out such dependencies would most likely lead nowhere or to inaccurate findings. Python’s matplotlib helped tremendously when trying to convey a message, summarize findings, or visualize variable behavior.

In general, MySQL is a much more optimal choice when it comes to pure data handling, storage, and retrieval. It excels in terms of both performance, security, and readability. Unfortunately, it does not allow for much more, which limits data exploration, profiling, and general analysis. However, these limitations did not make completing the task impossible, and the same JT value was obtained as in the previous task.