UNIVERSITY OF
WESTMINSTER
SCHOOL OF
COMPUTER SCIENCE & ENGINEERING

Data Matching for Southwark Council

by

HARSH KADIA

(W1709865)

Supervised by
DR. PHILIP WORALL

Submitted in partial fulfilment of the requirements of
the School of Computer Science & Engineering
of the University of Westminster
for award of the Master of Science

NOVEMBER 2019

DECLARATION

I, Harsh Kadia, declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This has not previously been accepted as part of any other degree submission.

Signed : Harsh Kadia

Date : 04/11/2019

FORM OF CONSENT

I, Harsh Kadia, hereby consent that this Project, submitted in partial fulfilment of the requirements for the award of the MSc degree, if successful, may be made available in paper or electronic format for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : Harsh Kadia

Date : 04/11/2019

Abstract

Data matching across the three databases of different departments of the Council can provide benefits by finding households and the people that are disadvantaged of the three departments. In order to reduce the number of people not getting an advantage of the available benefit system approved by the government of the UK by the Councils across the nation, the data matching should be implemented to get insights. All three databases are merged within one dataset and then are analyzed by the similarity of the scores based on the probability match of the records. The entities of the database to be matched are shuffled and integrated using the Associate Operation of math in order to find out the highest similarity record pair with the most reliable associate operation using intersection across three distinct databases. Several real-world problems have been considered based on the dataset and the future work to be done for the Council in upcoming years. Finally, quite a few recommendations for the broader use of the database are made to resolve possible disputes.

Keywords:

Southwark Council, Data Matching, Mosaic, Synergy, CapitaOne, Fuzzy Lookup, Children Social Care, School Census, Children Centres

Acknowledgements

I would like to thank Southwark Council to allow me to use their data. In particular, the Data Team and the IT team, for their support in ensuring that I had a comprehensive understanding of the data.

I would also like to thank to the supervisor, Dr. Philip Worall, for his guidance through the delivery of this project.

TOTAL WORD COUNT

(Excluding cover page, abstract, acknowledgements, table of contents, references, and appendix)

10,529

(Maximum without penalty: 16,500)

Table of Contents

1. Introduction 4

2. Literature Review 7

3. Scope and Objectives 19

3.1 Aim 19

3.2 Objectives 19

3.3 Data 19

3.4 Team Workflow 21

3.4.1 Mosaic (Management Information System): 21

3.4.2 Synergy (Management Information System): 21

3.4.3 Capita One (Management Information System): 21

3.5 Problem Domain and Constraints 22

4. Data 23

4.1 Data Collection 23

4.2 Size 23

4.3 Security 23

4.4 Project Planning 23

5. Methodology 24

5.1 Proposed Techniques 24

5.2 Implementation 24

6. Results 28

6.1 Primary Attempt 29

6.2 Final Attempt 29

6.3 NHS number Fetching 30

7. Conclusion 31

7.1 Potential improvements to the database 31

7.2 Further use of Methodology at the Council 31

8. Further Work 33

9. References 34

A. Appendix 37

1. Capita One (School Census) 37

2. Mosaic (Children Social Care) 38

3. Synergy (Children Centres) 40

4. Project Pre-Plan 40

5. Mosaic SQL Code 42

6. Match 61

7. Rstudio code 64

1. Introduction

Local government is responsible for a variety of vital services to individuals and businesses in specified communities. These provide well-known services such as social care, education, accommodation, infrastructure, and waste disposal, but also less well-known functions such as insurance, business support, and registration systems. It mainly focusses on safeguarding the people.

Children and young people are generally subject to the Care Order and the task of the Children's Social Worker is to meet the duties of the Council as a corporate mother. Adult Social Care is a function of offering social support to help people live their lives. It's about empowering people to preserve their sovereignty and integrity.

The data and the workflow of three different systems of the Council on which the data matching is to be done are as follows:

1. Children Social Care data – Mosaic (Management Information System)
2. School Census data – Capita One (Management Information System)
3. Children Centres data – Synergy (Management Information System)

Data matching is to be done across all those is to get a more holistic view of the person and household. The main benefit of the matching is that one can find households, families' people that are disadvantaged by matching multiple datasets. For example: In adding a dataset, one could see who is considered as a child in need looking at the mosaic, when they match it in education database they could see who has got attendance issues. So it is about building up a bigger picture that is happening with that child or that family about the situation and addressing them.

Southwark Council was outsourcing this data matching to a private firm named as Xantura from 2015. Xantura company was responsible for doing the data matching and then indicating to the Council that when one of the families hit one of the six troubled families criteria which is mentioned in the Objective part of the dissertation.

The Southwark Council's contract with Xantura cost £62,000 per year for 2 contracts. The first contract was for data matching and the second was a management service where they do all the statutory returns of the various departments of the Council. So, the cost of the initial matching of the data was £32,000. They used various kinds of tools for data quality like DB Forge studio and xlcompare.

The program named Troubled Families is run across England for the families suffering from various difficulties like domestic abuse, unemployment, absence in the school for children, anti-social behaviour, and mental health problems.

To identify those types of families in the local area, a local authority is assigned which acts as a single point of contact. Local authorities are paid by the Central Government for the families that meet the criteria. There were two phases of this program by the Central Government of England. The first phase was between 2012 to 2015 where the local authorities worked with approximately 120,000 families for which £448 million was allocated for those families.

The second phase started in 2015, which is to be continued till 2020, is hoped to help 400,000 families for which the Central Government has allocated £920 million. Also, in the second phase, more problems were added to the list for the families suffering from health, mental, drug abuse, domestic violence and children at risk.

The body named Southwark Council has to look after those families of the Southwark region which are considered under troubled families. Currently, Southwark Council has three different databases for troubled families. So, by matching all those three databases into one with the highest accuracy, it would be adequate for the employees to get an insight into the data and make a decision based on the matched criteria.

Troubled Families matter is aimed to enhance the outcomes in various domains like education, physical and mental health care and to lessen the requirement of statutory intervention through child protection. Those Troubled Families matter is implemented via a tiered model ranging from children to families with little or no additional needs to those who require Children's Social Care comprehensive support. In order to accomplish that, there is a continuity of care which guarantees that those who help at different intensity levels to provide a consistent service and that support continues to be given to those who no longer needs a statutory intervention (e.g. Children who no longer belongs to a Child Protection Plan).

Also, there is a Multiagency Threshold Guide published by Southwark Safeguarding Children Board which states that it has been developed to support and elevate the early and effective way to identify the needs, and to cooperate professionals by deciding the best ways to protect children, young people and families.

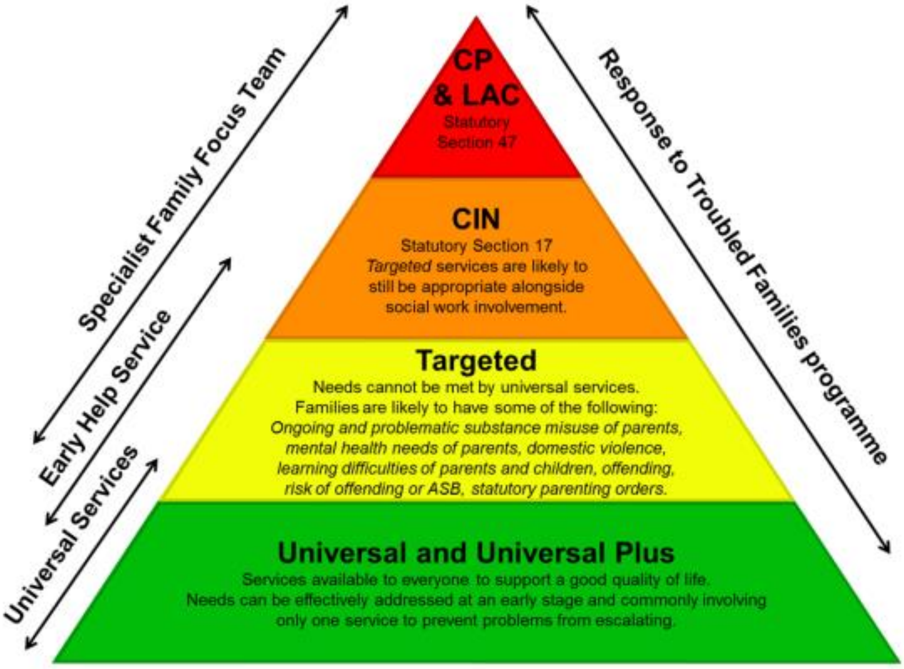
Based on the levels of the complexity of the issues of the problems, a decision had been made by the authorities that the low-level requirements can and should be accomplished within the universal provision. Whereas, an additional targeted acknowledgment via early help arrangement for the higher-level standard issues. Troubled Families Matter guarantees that at each tier of support, there is a range of services available. Below is the information on those four levels:

1. Level 1: Universal and Universal plus services – Children, young people, and families will access the universal services as and when required within single agencies. A gist of some of the services provided at this level is children centres, housing, police, GP and community health, etc.

2. Level 2: Targeted early help services – Some of the early health services that are provided at this level are Specialist family focussed, the team around the family interventions led by the other agencies, etc.

3. Level 3: Children In Need – A MASH Interagency referral form should be completed with the child which will help to identify their strengths and needs as well as gain specialist support from a child’s special care.

4. Level 4: Child Protection – The Universal and early help services will be fully involved in the case which includes through membership either through a group of professionals or the Child Protection Conference. Different agencies may lead to different aspects of the case.


Specialist Family Focus Team
Early Help Service
Universal Services
Response to Troubled Families programme
CP
& LAC
Statutory
Section 47
CIN
Statutory Section 17
Targeted services are likely to still be appropriate alongside social work involvement.
Targeted
Needs cannot be met by universal services.
Families are likely to have some of the following:
Ongoing and problematic substance misuse of parents, mental health needs of parents, domestic violence, learning difficulties of parents and children, offending, risk of offending or ASB, statutory parenting orders.
Universal and Universal Plus
Services available to everyone to support a good quality of life.
Needs can be effectively addressed at an early stage and commonly involving only one service to prevent problems from escalating.

2. Literature Review

Data matching is defined as a segregation of the datasets to conclude the matching results which include recognition of patterns and trends. The main aim of data matching is to analyse the inconsistencies that may be a fraud. Although it is obvious that it can't be used to analyse the patterns and trends in the behaviour of the data.

There are mainly five steps for the data matching process. The first step is data pre-processing, which assures that both the data are in the same format. The second step is the indexing which reduces the complexity of the process of data matching using data structures that provides effective and efficient generation that corresponds to the records from both the databases.

The third step is the comparison of the records where the record pairs comparison is done. Those pairs are then compared by distinct fields and record comparison functions. Next is the classification step, where pairs are allocated into matches, non-matches and partial matches which also depends on the method and the software used for data matching. Last is the evaluation step, where the quality of the matched data and completeness of the matched data are evaluated.

Data pre-processing is very essential to ensure the completeness of the dataset in a normalized and standardized structure. It is most important when the records have been added to the dataset concurrently and with the changes which are necessary for data entry methods which are also known as validation of the dataset. The indexing step is also necessary to check for redundancy because comparing each and individual record in a dataset with another dataset might have a computation complexity.

The flowchart for the two databases to be matched is shown in [**Fig. 1**]:

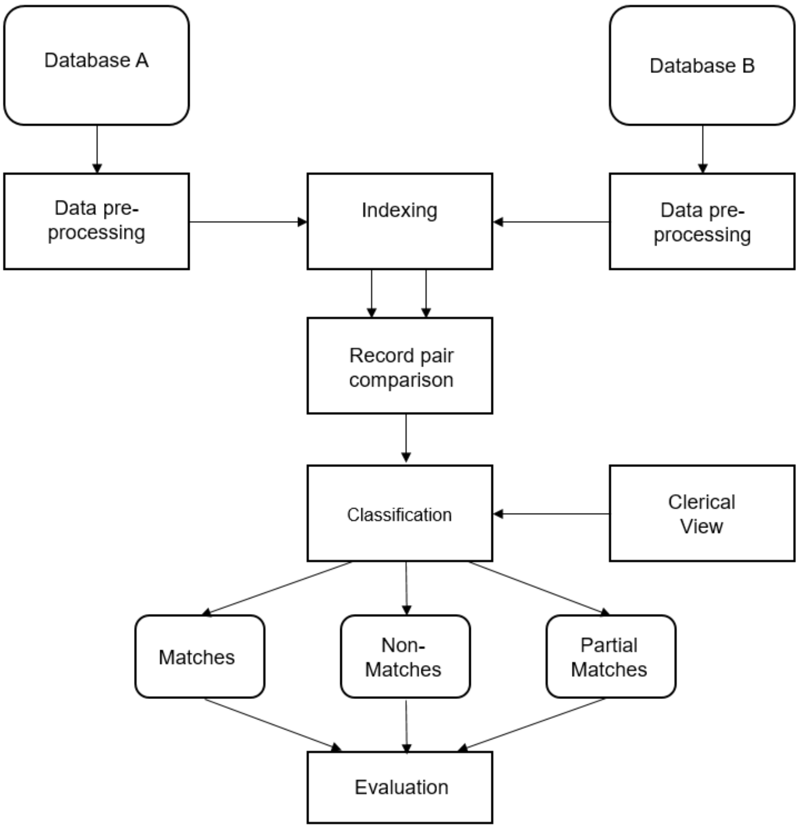
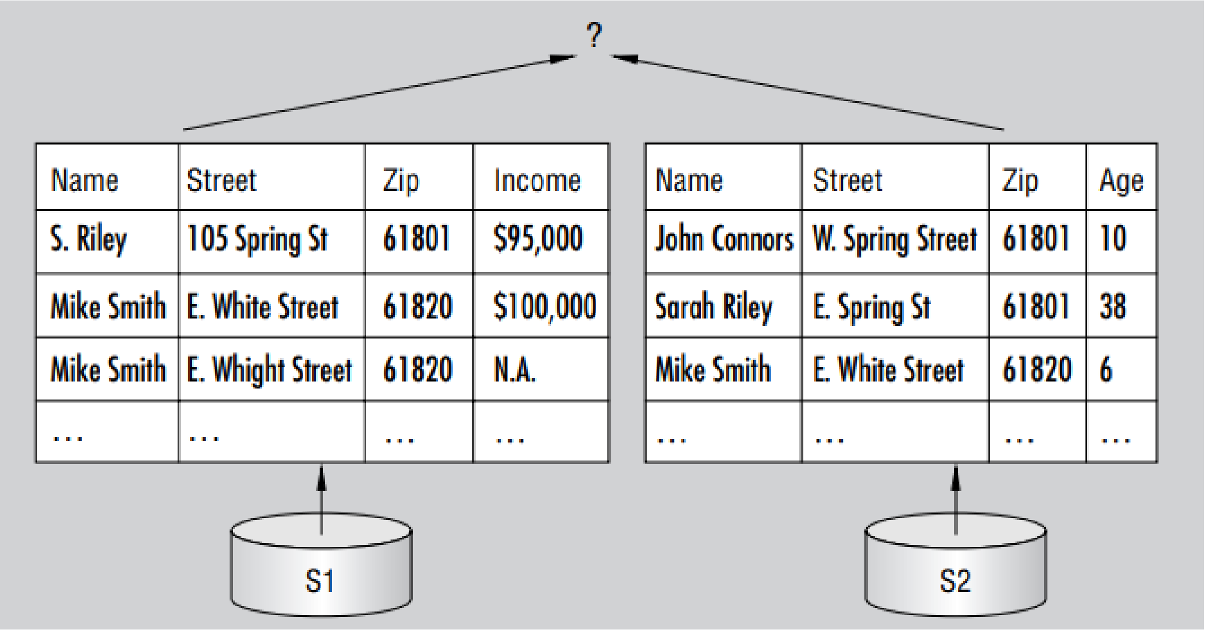


Fig. 1: Flowchart for data matching

One of the research papers from IEEE Explore named Profile-based Object Matching for Information Integration (Doan et al., 2003) describes how their researchers came across various object-oriented matching solutions in both the fields of database administration and AI. Basically, the solutions they came across were assumed that the target entities or records share the same set of attributes and they match the records by comparing the rows using similarity.

For instance, a government office wants to merge two departments and their databases namely (S1 and S2) about the people residing in Champaign and Illinois. **Fig. 2** shows the information on a single person. Since the database has the same geographical area, they have many duplicate rows. For example, S1's row S. Riley, 105 Spring St, 61801. $ 95,000 indicates the same identity of the person as S2's Sarah Riley, E. Spring St, 61801, 38 wherein the common attributes are name, street and zip code, whereas the different attributes are income in S1 and age in S2.

Their key to profile-based object matching solution was that the disjoint entities are correlated and can be able to execute a sanity check for the database matching. By the attributes matching from S1 and S2, only income and age are not matched. The rows of S1 row **Mike Smith, E. White Street, 61820, $100000** and the DB2 row **Mike Smith, E. White Street, 61820, 6** would show a match but practically this is very unlikely to be true because when they combined the two rows they came across a solution that "Mike Smith," a six-year-old with a $100000 yearly income.



Name	Street	Zip	Income
S. Riley	105 Spring St	61801	$95,000
Mike Smith	E. White Street	61820	$100,000
Mike Smith	E. Whight Street	61820	N.A.
...	...	...	...

Name	Street	Zip	Age
John Connors	W. Spring Street	61801	10
Sarah Riley	E. Spring St	61801	38
Mike Smith	E. White Street	61820	6
...	...	...	...

Fig. 2: Integration of S1 and S2

To understand their PROM system also known as Profile-based Object Matching, let's take an example of with two different relational tables: one consists of information about movies and the other table consisting of the review of movie (shown in the **Fig. 3**) Here, pyear, ryear and rrating means produced, reviewed and rating of the reviewer's movie, respectively.

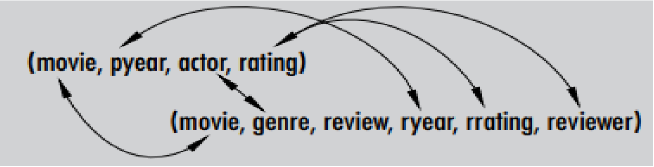

(movie, pyear, actor, rating)
(movie, genre, review, ryear, rrating, reviewer)

Fig. 3: Schemas of the movie domain

Now, their PROM system matches by their system of object matching technique and if the similarity is low then their system ignores the match by discarding the pairs as non-matched. Otherwise, it performs the sanity checks from other profilers to the pairs of the row.

Their PROM system lets the user to develop and transfer the matching tasks (in the terms of profilers) to match across various tasks and also, helps to improvise match accuracy further down by providing a framework into which users can input the newly developed profilers.

Fig. 4 shows their PROM system architecture in which the similarity estimator checks for a value from the input data and checks if it's a match or not from the two tuples T1 and T2. It solely checks on the base of shared attributes. If the estimator checks for the value of similarity which is low then it discards or else it passes it onto further matching the filter.

There is a set of hard profilers which contains hard constraints. For instance, a hard constraint for a movie would be that the review year must not be more than a year from the day the movie was produced. Another example for movie actors would be a particular actor not acting in a movie that has an average rating of less than 4. Hard profilers can be developed manually as well as automatically also by studying the complete data and also assuming that the data is complete.

They have a set of soft profilers as well as how well their rows match their profile on the basis of confidence scores. It is almost the same as hard profilers but with a soft constraint that it will satisfy the instances to be matched. Let's take an example soft profiler for a movie that may determine IMDB's rating and rating of Ebert are in correlation with each other when they differ from less than 3 (most of the movies satisfy this criterion). Soft profilers can be developed manually by the domain experts and other users, and then check the input data for the confidence score. Also, it could be created by train dataset by using the Bayesian network for an IMDB's movie instances on the basis of domain data.

Since hard profiler pulls out yes-no predictions and soft profiler give confidence based scores, they separate the amalgamation of two types of profilers. Also, they have the combiner which handles soft profiler and the matched filter handles the hard profiler, they believe that they get an improvement in the accuracy of the match by separating the methods. The matched filter uses the combination of AND to combine hard profiler's predictions. If the prediction is no by a hard profiler the overall result is no. The combiner has to merge the soft profilers' predictions by calculating the total of weighted confidence scores.

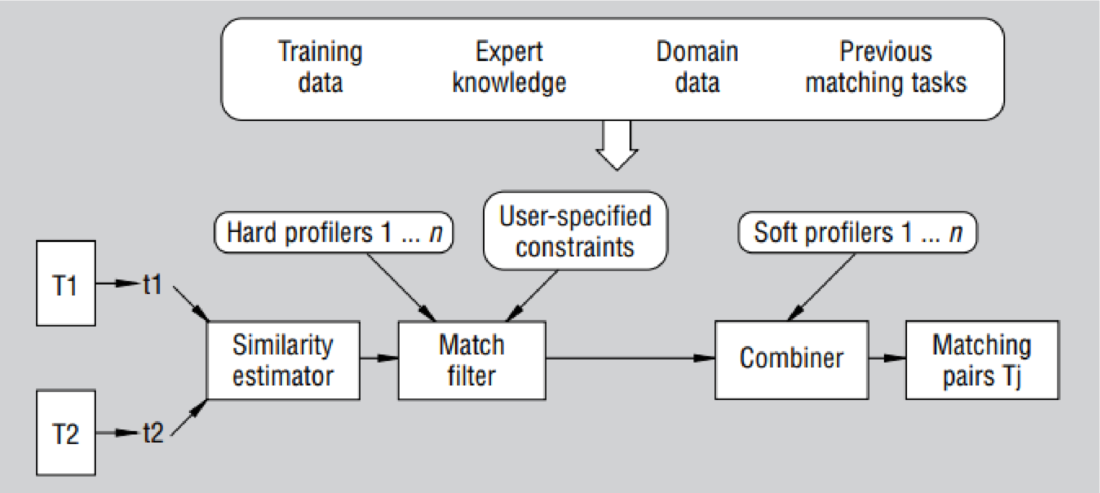


Fig. 4: The PROM system architecture

Their related work includes a plethora of solutions that were developed by the researchers in the field of the database, Artificial Intelligence, and data mining. Previously, manually decided rules were used to match the objects, whereas ample of them have matching rules form the train data of the input data tables. Efficient techniques were focussed to match the strings by several solutions, while other address techniques for scaling up the numerous objects. By comparing their shared attributes, the objects were matched.

One of the solutions was to add a layer to the attributes which are disjoint for maximizing the accuracy of the match. After studying thoroughly by the Artificial Intelligence research team, the knowledge reuse and the incorporation of the prior knowledge, they have considered reusing classifiers that are trained in other domains.

There were two ways that were different from Artificial Intelligence. Firstly, they reused knowledge types rather than classifiers (i.e. using manual profilers). Secondly, they developed task-dependant classifiers rather than arbitrary classifiers from other domains. Recently, researchers of the database are interested in reusing knowledge and are investigating on data integration and schema matching. It was their first work that attempted to reuse knowledge in the object-matching context.

A book named Linking Data for Health Services Research: A Framework and Instructional Guide (Dusetzina et al., 2014) which is available online as an Internet Source for reading describes a bit about the methods for data linkage. They have described two main types of algorithms for the data linking approach namely deterministic and probabilistic. Both have been implemented successfully in (Li Q, 2011). There are many factors to be considered for selecting the best algorithm such as time, available resources, research question, quality and the quantity of the entities to be linked. There is a set of guidelines in which they have described those two approaches in the terms of data quality, data availability and the goals.

The deterministic Linkage method gives an idea of whether the data pairs Agree or Disagree on a set of identifiers. The outcome of this approach would be as either All or Nothing. Also, the status of this match can be accessed using either a single-step process or a multiple-step process. In a single step, all the records are compared at once on an entire set of identifiers. If two records agree on the basis of character for character, on all identifiers and uniquely identified record pairs, then it is classified as a match. In a multiple-step strategy (also referred to as an iterative or stepwise strategy), records are matched in a collection of gradually much less restrictive steps in which record pairs that do no longer meet the first phase of criteria are exceeded to the second phase of match criteria for further comparison. If a pair meets the standards in any step, it is classified as a match. Otherwise, it is categorized as a nonmatch. Those approaches of deterministic linkage are also known as "exact deterministic" and "iterative or approximate deterministic".

National Cancer Institute used the iterative approach for creating the SEER (Surveillance, Epidemiology and End Results) has depicted high validity and reliability and hence has been successfully used in the SEER-Medicine dataset. It has two rounds for deterministic matching. In the first round, two records should be matched on either of those three listed:

1. First name and Last name
2. Last name, a month of birth, sex
3. First name, a month of birth, sex

The conditions in which partial or full identifiers are available but also may longer be transmitted, a deterministic linkage method can be applied. The deterministic approach does not consider the fact that various values have more discriminatory power than others.

To check the discriminatory power of an individual identifier and the tendency that two record pairs are matched whether they are matched or nonmatched, the probabilistic approached has been developed.

As per the model created by Fellegi and Sunter, matched pairs can be counted on the basis of linkage scores and the decision rules. Let's say that we have two files X and Y, where the file X is containing 100 records and file Y is containing 1000 records. By the Cartesian products of all possible pairs, 1000*100=100000 is the maximum possible match. Cartesian product is often impractical computationally when it comes to dealing with large files. In those conditions, it is desirable to lessen the space for comparison to the pairs that matches the criteria. It is also known as "blocking" subsets the large dataset into the small dataset with at least one common factor such as a specific condition or a geographic location. For example, the number of pairs that can be matched could be finite to only the pairs that match on the diagnosis of the clinical record or on a country of residence and month of birth. The weight appointed to agree or disagree on an identifier is based on a likelihood percentage score.

This theory can be applied to any of the identifiers whose value is distributed differentially. The calculation of an agreement based weight is counted by dividing the m- probability by the u-probability and the log2 of the quotient, when the two records are agreed on a particular identifier.

There is also a string comparator which reduces the probability of the match, like a length of the string, the number of transpositions of the characters, number of common characters, location of the string where there was a nonmatch. Also, short names would be assigned lower weights or when the first character of the string is not matched within the files.

For instance, the weight of a full name is 12 and the value of the string comparator is 0.95 of the first name matching with both the records of different datasets, the partial agreement weight would be counted as 12*0.95 = 11.4. The improvement in the matching can be done with the use of a string comparator when there is an expectation of typographical (manual/human) error. It can also be assessed by plotting the results of the linkage or the number of records that matched using a histogram.

If the algorithm is appropriate, then the plot would depict a bimodal distribution of the scores, where the one end would be the lowest scores for the data that resulted in a non-match based on the likelihood and at the other end the peak of high scores where the match has resulted in a very likely match. The cutoff threshold can also be defined depending on the nature of the study or on the research question.

Steps for the summary of using Probabilistic Record Linkage:

1. Assume *a* and *b* probabilities for the variable that is linking using the frequency of agreement and disagreement within all the pairs.
2. Calculation of weights for *a* and *u* probabilities for agreement and disagreement.
3. Calculation of a total weight linked for an individual pair by adding the weight linked of each variable.
4. To set a threshold and then compare which pairs are considered to be as linked based on the research question.

In the scenarios where the identifiers are available directly and also have good quality, deterministic methods are suggested to be used because they are easy in interpreting, implementing and efficient. While for the scenarios where the information is poor i.e. when te identifiers are not available or the quality of the data is poor, probabilistic methods are the most preferred and that outweighs deterministic methods that are beneficial for the goals of the project and extra time is available for the implementation.

So, the overall process can be seen in **Fig. 5**:

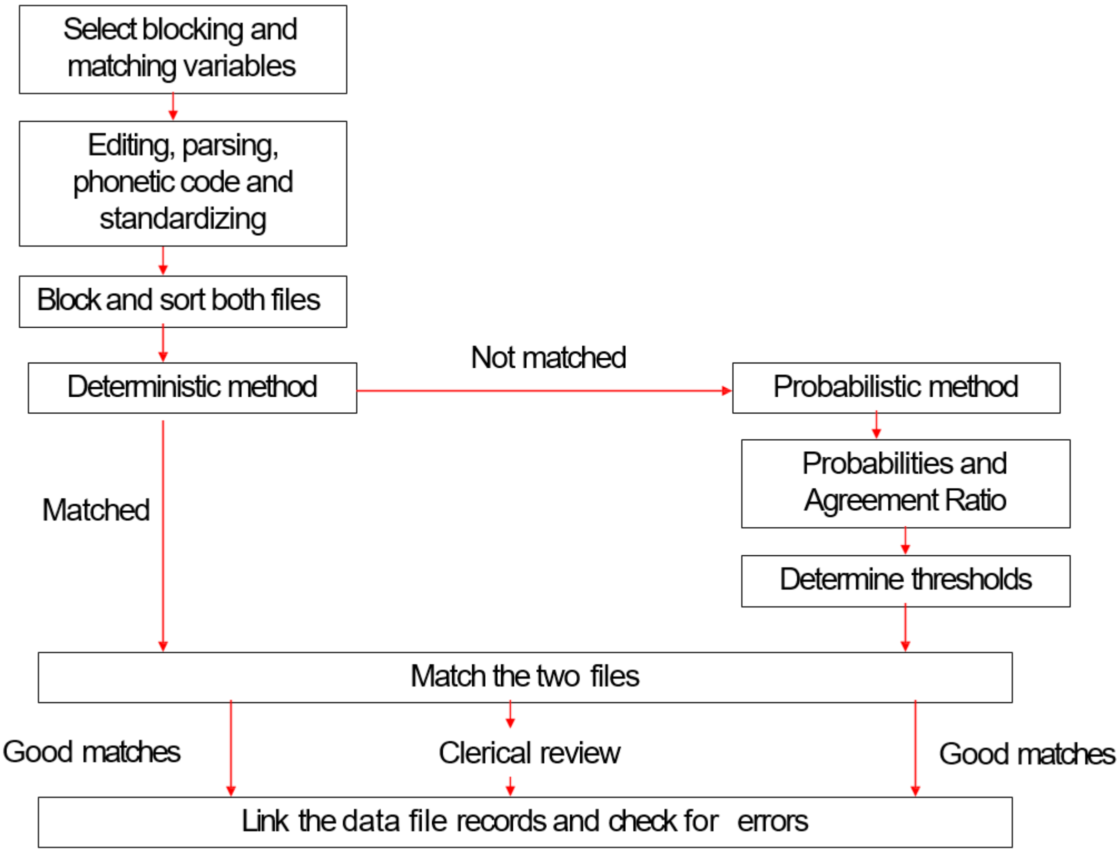

Select blocking and matching variables
Editing, parsing, phonetic code and standardizing
Block and sort both files
Deterministic method
Not matched
Probabilistic method
Matched
Probabilities and Agreement Ratio
Determine thresholds
Match the two files
Good matches
Clerical review
Good matches
Link the data file records and check for errors

Fig. 5: Overall Process of a Deterministic and Probabilistic method

One of the journal paper named A Fuzzy Logic Approach to Case Matching and retrieval suitable to SQL Implementation (Portinale and Montani, 2008) describes a fuzzy logic case study for retrieval and matching similarities. They presented a method of Local approval of a function that can be represented by a means of fuzzy distributions on its field, with the abstraction of actual values to linguistic terms. In addition, global acceptance is based entirely on fuzzy logic, using the normal combinations of local distributions by the way of strictly specifying standards. They suggested a framework for retrieval through fuzzy SQL methodology.

The matching algorithm and the case extraction usually focus on the application of the Nearest-Neighbour (NN) approach. To obtain a retrieved case, a combination of local distance metrics is compared to an individual feature between the query and the case (Mantaras and Plaza (1997)). A case retrieval algorithm k-NN will then return the k nearest cases to the target one, assuming that the resemblance of the two cases is the opposite of their length, so that the k cases which are nearest to the target ones are the most similar. The topic of approximate matching (so important to CBR) has historically been dealt with using fuzzy techniques in several AI subfields other than CBR. Such approaches to CBR can be transposed by considering the definition of a fuzzy linguistic term as a "similarity dimension" on a case-by-case basis in order to compare the values of the feature. In addition, a linguistic parameter over a feature's domain can be used to define match acceptance (and thus retrieval) of cases with different (fuzzy) values for that feature. The underlying idea is that if two attribute values of membership are "next" to a specified fuzzy array to be used as a comparison example, then those values are identical to that context.

A second issue relevant to normal case retrieval concerns the reality that appropriate case structuring and case base arrangement need to be formed in order to implement a specific algorithm (Mantaras and Plaza (1997)). This may be an additional burden on the development of a CBR framework, particularly when case information is already available in standard relational databases, as is the case in many applications. There has been little effort to identify distance-based methods that specifically target SQL (Schumacher et al., 2000). In the past few years, on the contrary, they have supported a burgeoning interest over defining fuzzy additions to regular SQL, contributing to a variety of suggestions for defining a SQL-like language capable of dealing with fuzzy information for the data.

Their main purpose was to propose a concept of case matching and acceptance in a case retrieval system depending on the similarities of the characteristics using fuzzy logic. At the same time, they intended to suggest a specific retrieval framework focused on the above ideas and realized directly on a SQL engine by the means of a fuzzy extension of SQL.

For making their fuzzy case retrieval system successful on RDBMS, they have established the following model of implementation:

- Cases are interpreted as rows in relation, such type of relation can be observed by a table delineating the case base *CB*.
- The case characteristics are defined by generic attributes, simple SQL categories define the function class (eg. Int for a distinct linear feature, float for a continuous linear feature, varchar for marginal features, and many more)
- Fuzzy operations and terms are described by an adequate meta-database containing all the fuzzy information.
- A query case is specified by defining the values for a set of characteristics.
- The retrieval happens after the acceptability threshold λ, by creating a fuzzy SQL query with the threshold λ.

Once the case base CB has defined a request case q and an acceptability threshold λ, a fuzzy SQL query can be created as follows:

SELECT (λ) * FROM CB WHERE RCq

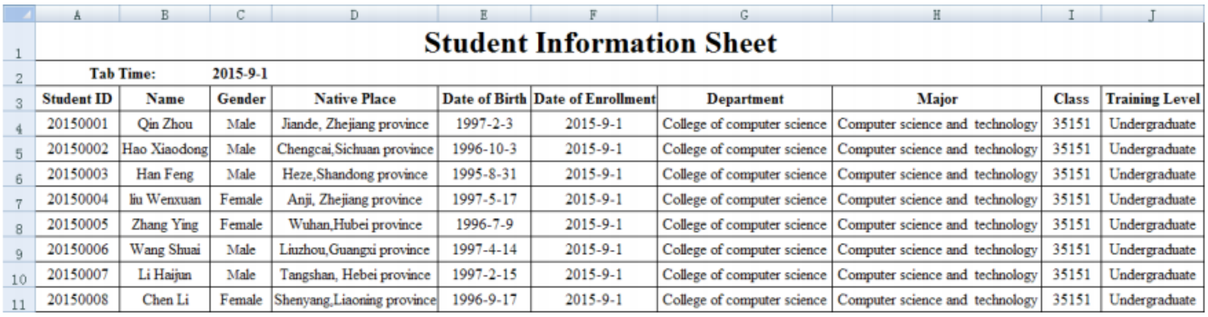
Retrieved cases are extended as the result table rows obtained from the query. Obviously, if one is involved in only one of the case features sub-set A, the query's target list will be A instead of *.

They have proposed a method in which local support and similarities to a function can be convoyed on its domain by fuzzy distributions. Global similarities and acceptance are then fully defined in fuzzy terms, using the local combinations by the way of clearly specified norms, and is thus entirely dependent on fuzzy logic. It is argued in [8] that fuzzy logic is admissible both in case retrieval and representation, where 'fuzzy data representation' and 'fuzzy matching' could be able to implement the appropriate tools. They concluded that the work into the tight relationship between CBR and fuzzy logic would eventually lead to the development of dynamic thinking structures that are capable of addressing problems of greater complications.

Talking more about the fuzzy lookup for MS Excel is the data integration space where it can be used for matching dirty textual data. Dirty textual data means spelling mistakes or the missing data (if you have a record it might not have some of the fields that has values) and the goal is to match is really efficiently because Microsoft and lot of other big companies have millions and millions of records that they need to deduplicate. So it is really trying to find the different representations of the same entity efficiently. The first step is to come up with a way to define the meaning of similarity between two textual entities. The second step is to efficiently index that. Let's say that you have an incoming record and you want to match it to your full list of products or your full list of customers and wanted to do that really fast. That was the challenge for the developers of fuzzy lookup plug-in for Excel. This technology was a part of the SQL server integration services but now they have moved it into space into excel so that everyone can use it in a more interactive way.

The article named The Application of Case Teaching in Excel Lookup and Reference Functions (CAI, 2017) begins with the development of a student card that incorporates the features of VLookup and Index and addresses the use of case teaching to Excel search and comparison features. Teachers instruct learners in their study, discussions, and application of the event in the learning process and the students take the initiative of learning yet sharing in order to achieve the educational aim and to understand the instruction that helps both teachers and students. In order to apply the query and index functions to Excel, a detailed case has been selected in the teaching process. This is the development of a student card which can make use of the vlookup, index and match feature to solve the individual problems and make the complicated work easy and orderly.

They had two sheets in Excel containing Student Information which contains all the information about the student (**Fig. 6**) and the other one is the Student Card Template (**Fig. 7**). Each student ID is different. Suppose the relative know-how of Excel could be used to retrieve the student information required from the student information sheet and to connect it to the student card in the model. By this, their task will be accomplished.



Student Information Sheet

Tab Time: **2015-9-1**

Student ID	Name	Gender	Native Place	Date of Birth	Date of Enrollment	Department	Major	Class	Training Level
20150001	Qin Zhou	Male	Jiande, Zhejiang province	1997-2-3	2015-9-1	College of computer science	Computer science and technology	35151	Undergraduate
20150002	Hao Xiaodong	Male	Chengcai,Sichuan province	1996-10-3	2015-9-1	College of computer science	Computer science and technology	35151	Undergraduate
20150003	Han Feng	Male	Heze,Shandong province	1995-8-31	2015-9-1	College of computer science	Computer science and technology	35151	Undergraduate
20150004	liu Wenxuan	Female	Anji, Zhejiang province	1997-5-17	2015-9-1	College of computer science	Computer science and technology	35151	Undergraduate
20150005	Zhang Ying	Female	Wuhan,Hubei province	1996-7-9	2015-9-1	College of computer science	Computer science and technology	35151	Undergraduate
20150006	Wang Shuai	Male	Liuzhou,Guangxi province	1997-4-14	2015-9-1	College of computer science	Computer science and technology	35151	Undergraduate
20150007	Li Haijun	Male	Tangshan, Hebei province	1997-2-15	2015-9-1	College of computer science	Computer science and technology	35151	Undergraduate
20150008	Chen Li	Female	Shenyang,Liaoning province	1996-9-17	2015-9-1	College of computer science	Computer science and technology	35151	Undergraduate

(**Fig. 6**: Student Information Sheet)

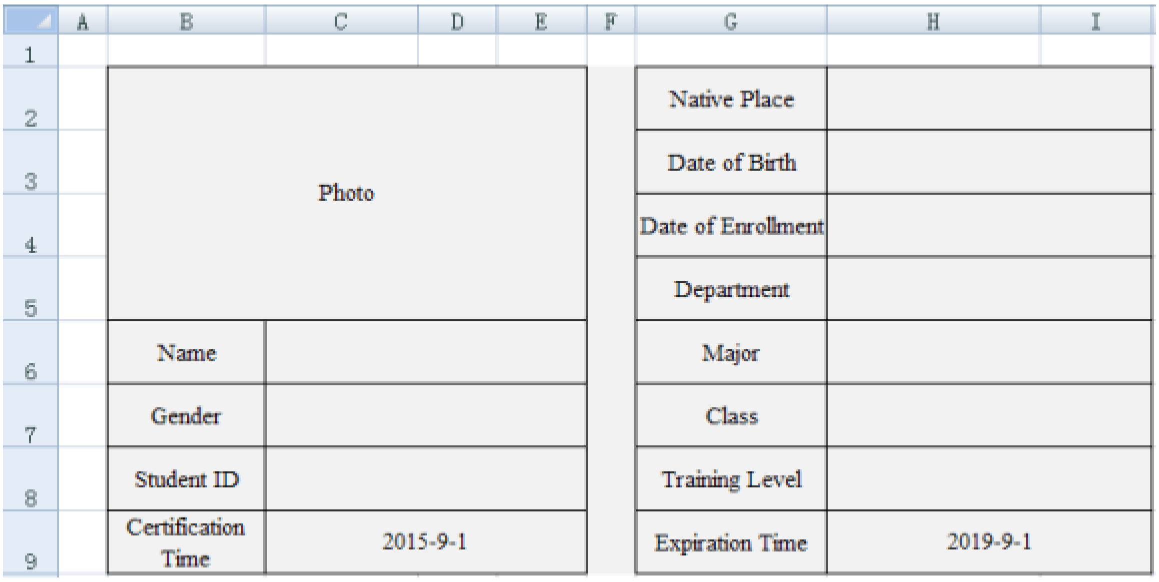


Photo		Native Place	
		Date of Birth	
		Date of Enrollment	
		Department	
Name		Major	
Gender		Class	
Student ID		Training Level	
Certification Time	2015-9-1	Expiration Time	2019-9-1

(**Fig. 7**: Student Card Template)

Automatic data filling in the student card model is based on the student ID. This task could be realized with vlookup, which identifies and extracts the relevant information in the student information file.

For instance:

According to the student ID 20150001, finding the name to be paired.

Entering the student ID 20150001 in the card ID as displayed in **Fig. 3**. Select C6 once in the student card template, select "formula" to "insert function," select "lookup and reference" in the "or choose category" window of the pop-up insert method, then select "vlookup" and select "Ok" and you can see the pop-up window of the vlookup function. Check for the stated value in the first column of the table array and refer to the position of the defined column in the current row of the table array (Li Rong, 2017).

It is quick for students to find a difference in study and work, and to learn actively. The willingness of learners to evaluate and solve problems has been developed by case analysis, discussions and training. This problem is an amalgamation of vlookup and index function. It is not only necessary to practice the abilities of students applying Excel lookup and reference functions to solve real problems, but also to improve the ability of students to analyze and act accordingly.

There are many data integrating software and data quality tools can be beneficial with a better user interface as compared to manually coding the matching. Software like Magic Quadrant for Data Quality Tools can be used for this project. Although there may be many types of software, selecting the best and effective would be a key.

The first and foremost challenges of name matching are Phonetic similarity (Jesus – Heyzeus - Haezoos), Missing Spaces & Hyphens (MaryEllen ↔ Mary Ellen ↔ Mary-Ellen), Missing Components (Phillip Charles Carr ↔ Phillip Carr), Split Database Fields (Dick. Van Dyke ↔ Dick Van. Dyke), Spelling Differences (Abdul Rasheed ↔ Abd al-Rashid), Titles & Honorifics (Dr. ↔ Mr. ↔ Ph.D.), Nicknames (William ↔ Will ↔ Bill ↔ Billy), Truncated Components (McDonalds ↔ McDonald ↔ McD) and Initials (J.J. Smith ↔ James Earl Smith) which are to be taken care of while matching the databases and their variables.

Some of the name matching algorithms which can be used are Common Key Method (Assigns names a key or code based on their English pronunciation such that similar-sounding names share the same key.), List Method (Generates a list of all possible variations in the field of each component name and then matches names from that list), Edit Distance Method (Calculates the lowest set of changes — in various ways — required to get from one name to another.), Statistical Similarity Method (Develops a statistical algorithm by training thousands of paired names to calculate the similarity score between two names.), Wrong Embedding Method (Turns each word into a numerical vector based on its semantic meaning and calculates the similarity of two words in a multidimensional space. Commonly used for organization names.) and Hybrid Method (Combines some or all of the name matching methods above.).

Moreover, Natural Language Processing can also be used for text matching. Fuzzywuzzy is a Python library uses Levenshtein Distance to calculate the differences between sequences in a simple-to-use package.

3. Scope and Objectives

3.1 Aim

The aim of this project is to develop an automated algorithm that will match the three different databases based on the given criteria with the highest percentage of the matched score, after which the data needs to be analysed and cross-checked with the original data.

3.2 Objectives

The first and foremost objective of this project is to understand and select the data which is necessary from three databases as it contains the criteria for Troubled Families. In total, there are six types of family issues that are set under these criteria (Southwark Council, 2018). The family issues are as follows:

1. Parents and children involved in crime or anti-social behaviour
2. Children who have not been attending school regularly
3. Children of all ages who need help, are identified as in need or are subject to a Child Protection Plan
4. Adults out of work or at risk of financial exclusion or young people at risk of worklessness
5. Families affected by domestic violence and abuse
6. Parents and Children with a range of health problems

Based on those six criteria, the data matching is to be done so that unnecessary columns in the database are not selected which would result in less computing time/query optimization. Only the needed columns need to be selected which makes sense for matching based on the percentage score from the different databases.

Secondly, the use of a data quality tool with an interactive user interface that is better than coding manually which would be time-consuming and prone to human error.

Finally, there should be an expected higher matching of accuracy as the algorithms should be able to deal more objectively with misspelled or similar names than with a manual human process. Hence, there should be something that can confirm if that entity is correct or not.

3.3 Data

There are three different Management Information Systems for Children Social Care, Children Centres and School Census which is stored in the Management Information Systems named

Mosaic, Synergy and Capita One. The observations of Mosaic, Synergy and Capita One are given in ***Appendix 1, 2 and 3*** at the end of the document.

There is a Unique Property Reference Number (UPRN) for each and every house of the Council where the people staying in the houses are registered and it is provided by an external organization named Ordnance Survey. However, not everybody that is in Mosaic is on Address Base. When they got the address base it was 2 years ago. New addresses put in the database will be in the address base format and the people registered before 2 years are registered in the old database which is a free text file. And that is the reason for much missing value in the UPRN.

They started outsourcing to Xantura in 2015. The initiation of Data matching to Xantura from a Social Care perspective came through the Troubled Families program Phase 2. They didn't have a holistic view of a child or a household, they only had the information about the Child Social Care provided by the Social Workers.

NHS number fetched to all the databases is supposed to go forward as a Unique Identifier for Everybody. Before there was only UPRN as a unique identifier for all, not now with NHS number it would be easy and non-redundant. As the problem with UPRN is that people might have moved to different places to stay and might not have been updated in the system's database and that would not be much help considering the scenarios to safeguard the people with the utmost care possible.

Every house in the UK has its own UPRN number and that's the reason for using Unique Property Reference, not the Pupil Number. So if I match 2 records it would match the house and not the person in it because the person may have moved.

When they get to the point where the NHS Number is the Unique Identifier, they would be able to search on Mosaic for the NHS number and with that, one could see if a person is receiving Social Care interventions or not and will be able to look at the history of that case. Also using the same identifier, Children Service or MASH (Multi-Agency Safeguarding Hubs) can access health records for the same person and see the health history of a person. If the NHS number is used in Schools then they can search the School's data. Hence, the unique identifier of NHS Number across whole systems is the future for having a long term benefit of the citizen.

Currently, it is used in Adult Services because there is something called the Care Act and from that one of the recommendation is to use the NHS Number as a Unique Identifier for all adults and that will make a way down to the Children Care. Children for the age of school have UPN (Unique Pupil Number) because it is similar to everybody having an NHS Number. Everybody of school age should have a Unique Pupil Number which is a part of Mosaic data currently and is a part of statutory returns. The department of Education matches the Social Care returns with the School Census information to find out CIN (Children In Need) with absence issues and attainment information through the use of Unique Pupil Number.

3.4 Team Workflow

3.4.1 Mosaic (Management Information System):

When the information comes into the Children Social Care it goes into the MASH (Multi-Agency Safeguarding Hubs) first, MASH collects information from the Schools, neighbors, families referring themselves, hospitals, etc. Also, the problem is that when a baby is born the Council doesn't get notified. MASH records it into the Mosaic system. The team then writes the reports that extract the data from Mosaic through Social workers. The end process is statutory returns. Every year and sometimes quarterly depending on what the returns are, quality and performance teams are responsible for submitting statutory returns to the department for Education (DFE). The performance team is responsible for the data quality and then submits to the DFE once it has been signed off. The director has to sign off their returns and then it goes off to the DFE. This whole process of extraction is known as the Statutory return process. Also, monthly reporting is done on recent activities like the number of children in the Child Protection Plan, Number of assessments, and so on.

3.4.2 Synergy (Management Information System):

Synergy is the system where the Children Centres data is stored. Admin uploads the data to the Synergy and then the evaluation officer extracts the data to perform an initial analysis of the data stored in the Synergy. After the analysis, it is then sent to Public Health Strategic Hub leads for each locality to scrutinize the data and improve the highlighted areas. After then, the Children's and Adult's Board informs the Under 5's partnership to evidence the impact on the cohort.

3.4.3 Capita One (Management Information System):

First of all, the system administrator adds the raw XML data file into the Capita. Then the Education Data team will then process it in terms of matching and dealing with suspense items (In Capita, there is a number of categories on which they want to accept the data and overwrite in Capita or to put the file into the suspense and then the education data manager and their assistant will physically lookup the conflicts of data and decide which one should stay within the Capita). After then, the Education Data team receives the data files from the schools and then system admin will upload them into the COLLECT (COLLECT is a system/website called https://services.signin.education.gov.uk/ for the Department for Education which is only accessible to the Education Data Manager and an assistant with their login credentials). Once all of those are finalized then they will send the files to the admin to upload it into the Capita.

Data matching would be helpful for Data Educational Manager as she would be able to see who is involved with them currently with their current address and even with their current school. School Census is uploaded termly, whereas Mosaic and Synergy are live. So, if all the information is available in Capita, then it would be beneficial to the staff members as they would be able to see where those Pupils are at the moment whose children are registered in Children In Need rather than waiting for the next term to look into the Capita.

3.5 Problem Domain and Constraints

There are many missing values in the unique identifier named UPRN (Unique Property Reference Number) because not everyone in Mosaic is at the Address Base file provided by Ordnance Survey. It was 2 years ago that we received an address Base data. The latest addresses in the registry will be in the system, and the users listed for 2 years will be recorded in the existing list, which is a free text archive.

They also started NEET (Not in Education Employment or Training), SEND (Special Education Needs Disability) and EHC (Educational Health Care Plan) to evaluate children between 16-18 age who are not in University or college nor in an employment.plan 2 years ago recording Educational Health and Care plan on Mosaic in a certain area but it has not been kept up at the moment due to the lack of consistency of the data and many more data quality issues which could not provide much help for the improvement.

Also, there was a limitation of using software tools such as Rstudio, Python, etc. since I cannot have the data in my own system and have to use the systems provided to me within the Council. I had to do in their provided systems only which would restrict me to use only Oracle SQL Server or MS Excel.

I opted to do it in MS Excel since everyone can get access to the file and would be easy for a person having the least experience with coding to understand and manipulate it according to his/her needs as per the requirement. The problem with SQL was that one would not easily open the document and able to see the data matched across all systems. If one department would have a SQL developer, there are chances that others would not have a SQL developer and would be a hindrance for the other department to access the file. Hence, for the betterment for all the departments, I was suggested to do it in MS Excel from the future point of view.

Also, many of the taxpayers argue that they have already spent a lot on these families, why is the government spending more and the local government is determined to get the results and value for money that will save money for the taxpayer in the end. Those are the difficult challenges for any government.

4. Data

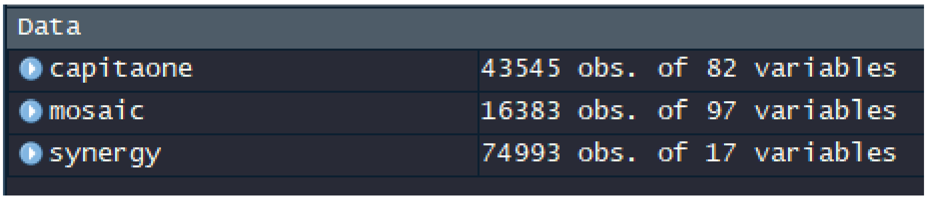
4.1 Data Collection

First of all, the data which was stored into the Council's Management Information System named Mosaic, Synergy and Capita One was extracted using Oracle SQL Server 19.1.0. After extracting the raw data the file is then converted into .xls or .xlsx file so that it can be used in MS Excel for the data pre-processing and proper formatting.

The data from each department contains various files of it. So to identify the best one with the appropriate and relatable data, the latest and the better data quality file is to be chosen in order to get the best results matched across the systems.

4.2 Size

An on-going analysis is rather large based on the potential dataset of the Council. The size of the data that is to be matched is as follows:



```
Data
capitaone        43545 obs. of 82 variables
mosaic           16383 obs. of 97 variables
synergy          74993 obs. of 17 variables
```

4.3 Security

The data used as a part of the matching process includes the information of the entire Children and Social Care, School Census and Children Centres and therefore has a range of important fields that need to be handled carefully. Any misuse of this information may impose a threat to Council's data protection credibility, hence connecting data back to the origin of the accounts will be carried out on aliases or even on Identity fields that have little or no significance to those outside the organization.

4.4 Project Planning

I found the opportunity of the project in the Networking event of the University where many delegates appeared and explained about their company's prospectus and projects they're currently working on. I contacted one of the champ who used to work at Southwark Council by approaching him on emails and it was the final decision that was made on 29 April 2019 to start the project according to the availability. After then, few files were shared for my research by the Council and then I started working on it from 3 June 2019.

In order to accomplish the proposed work of data matching across the three distinct systems of the Council, the project needs to be well organized, scheduled and detail-oriented. A pre-plan was created initially for the tasks which need to be done using Oracle SQL Server with the count of days to be assigned for a particular task. However, this pre-plan was not followed using Oracle SQL Server but instead the task assigned was accomplished using MS Excel which saved time and would be beneficial for the employees of the Council with the least know-how of SQL. The project pre-plan is attached in ***Appendix 4*** at the end of the document.

5. Methodology

5.1 Proposed Techniques

From the techniques proposed in the literature review and the work they have done, using SSIS also known as SQL Server Integration Services by Microsoft was the most preferable one. However, due to the limitations of the software tools available I had to implement using Oracle SQL Server and then I used MS Excel after the approval from one of the managers to whom I was reporting my work. We discussed the functioning and the working of Fuzzy logic tools and then he was happy to proceed with that plug-in for MS Excel as we would be able to cross-check it using spreadsheets and would be preferable for everyone.

The function of the Fuzzy Lookup is to provide a fuzzy analysis for textual data in MS Excel. We can use this plug-in to clean up challenging problems such as weeding out ('fuzzy matching') duplicate rows within a single table where the duplicates are actually duplicated but not matching exactly between two distinct tables. This plug-in is exceptionally efficient, particularly for a person who is used to match the functions like VLOOKUP in MS Excel. Fuzzy Lookup plug-in is available to download from https://www.microsoft.com/en-gb/download/details.aspx?id=15011.

5.2 Implementation

My primary task was understanding the systems and their functions in their own departments and based on the first name, last name, date of birth, postcode the matching was to be done. But before that, some of the tasks which need to be done were converting first names, last names and full names into the BLOCK (Capital) letters using Oracle SQL Server.

First of all, I started learning the SQL advanced codes from w3schools.com for about 2 weeks. Where in I have used SQL functions such as Inner join, rank, dense rank, left outer join, nvl, cross join, coalesce, union, order by, update, index, alter table, count, and many more. Moreover, data profiling, data quality, and master data management were needed in order to get complete insights for the data matching.

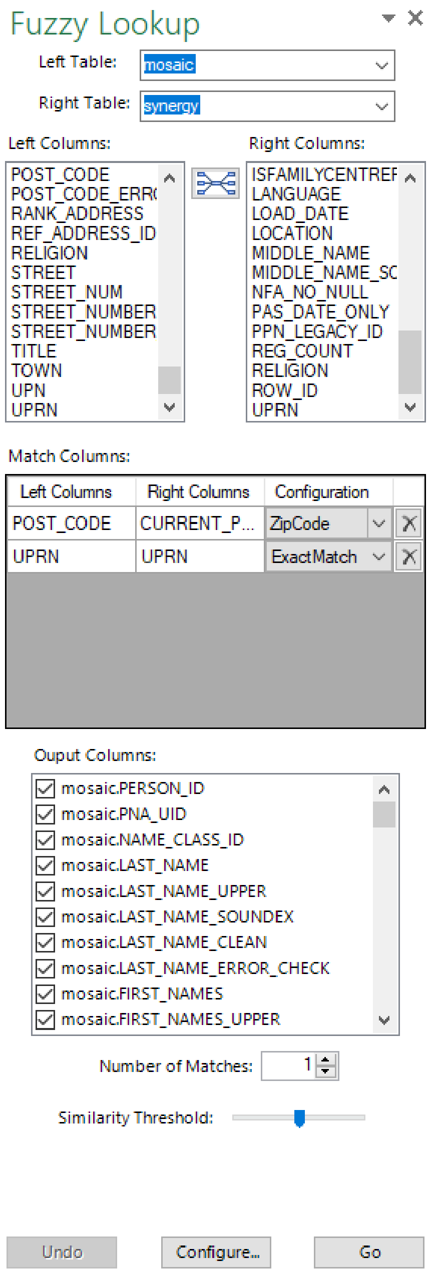
The SQL code and the output for pre-processing and joining the Mosaic dataset are attached in ***Appendix 5*** that includes sorting of most recent dates based on Closure_Started_On and Closure_Completed_On of the observations FEH (Family Early Help) and LAC (Looked After Child) as well as ranking it uniquely like dense rank and rank for FEH, Legal Status, and LAC. Also, the code and output of matching across the systems using the unique identifiers such as the first names, last name, post-codes, date of birth, first name Soundex, last name Soundex, UPRN (Unique Property Reference Number) are displayed in ***Appendix 6***.

Moreover, in some of the columns, the date needs to be ordered in either ascending or descending order and ranked in order to find the redundant number of individual identity of a person. Hence, the rank and rank DESC/ASC functions are used.

After then, I started working on MS Excel using Fuzzy Lookup plug-in which is similar to join operations in SQL, Vlookup in Excel and also shows the similarity match between the records of the tables that are compared.

For the pre-processing task of the three datasets, I have trimmed all the white spaces using the Trim function in excel so that the unwanted white spaces could be removed. After then, converted some of the rows such as date of birth to date format, UPRN, UPN, Person_ID to number format, and First names, Last names, full names converted to BLOCK letters and then formatting as general. Also, the count of records in UPRN was done by using COUNTA function and the duplicate values and null values were done using COUNTIF function and Remove Duplicate toolbar from the Data menu in MS Excel.

For matching those three datasets, all the datasets needs to be converted into table format which can be done by Ctrl + L. The fuzzy lookup plug-in needs to be installed into the system and after clicking on it, there appears a horizontal pane to the right side of the screen of MS Excel as displayed in **Figure 8**. Here only two datasets can be merged at a time and the left table will be table 1 (1st dataset) and the right table will be table 2 (2nd dataset). Also, similar to right join and left join operations in SQL. Thereafter, the columns of the left and right table are to be selected (for example UPRN from the left column and UPRN from the right column) and configured from five options of ZipCode, Exact match, Phone number, Social security number or default. Then in the output columns, one needs to select the columns that need to be displayed when the fuzzy matching is done. In the end, you select the similarity threshold of the similarity index. But for my project similarity of more than 90% is appropriate and the data which is between 90% to 99.87% could be modified for the next matching so that the data quality is improved.



Fuzzy Lookup

Left Table: mosaic

Right Table: synergy

Left Columns:

POST_CODE
POST_CODE_ERR(
RANK_ADDRESS
REF_ADDRESS_ID
RELIGION
STREET
STREET_NUM
STREET_NUMBER
STREET_NUMBER
TITLE
TOWN
UPN
UPRN

Right Columns:

ISFAMILYCENTREF
LANGUAGE
LOAD_DATE
LOCATION
MIDDLE_NAME
MIDDLE_NAME_SC
NFA_NO_NULL
PAS_DATE_ONLY
PPN_LEGACY_ID
REG_COUNT
RELIGION
ROW_ID
UPRN

Match Columns:

Left Columns	Right Columns	Configuration
POST_CODE	CURRENT_P...	ZipCode
UPRN	UPRN	ExactMatch

Ouput Columns:

☑ mosaic.PERSON_ID
☑ mosaic.PNA_UID
☑ mosaic.NAME_CLASS_ID
☑ mosaic.LAST_NAME
☑ mosaic.LAST_NAME_UPPER
☑ mosaic.LAST_NAME_SOUNDEX
☑ mosaic.LAST_NAME_CLEAN
☑ mosaic.LAST_NAME_ERROR_CHECK
☑ mosaic.FIRST_NAMES
☑ mosaic.FIRST_NAMES_UPPER

Number of Matches: 1

Similarity Threshold:

Fig. 8: Fuzzy Lookup

By Fuzzy Lookup approach, the similarity match between Mosaic and Synergy is as follows:

Similarity by the First name	Count of Records
100	26330
70-75	309
75-80	389
80-85	2382
85-90	3639
90-95	6240
95-99	721

Similarity by Last name	Count of Records
100	19956
70-75	396
75-80	595
80-85	3093
85-90	4545
90-95	6774
95-99	792

Similarity by Full name	Count of Records
100	754
70-75	280
75-80	360
80-85	404
85-90	555
90-95	893
95-99	335

Similarity by Date of Birth	Count of Records
100	539
70-75	0
75-80	173
80-85	904
85-90	19
90-95	79
95-99	954

Those were the major matches to be looked at, but also there were many other matches such as First name Soundex, Last Name Soundex, Postcodes, address, etc. which resulted in very little similarity and would not really help much in the data matching between Mosaic and Synergy datasets. Hence, by reporting this trail match analysis to the reporting manager, he suggested matching on the basis of a Unique Identifier by taking into account the UPRN (Unique Property Reference Number).

The initial attempt of matching all the three datasets was using the associate law such as (Mosaic ∩ Synergy) ∩ CapitaOne, (Mosaic ∩ CapitaOne) ∩ Synergy, and (Synergy ∩ CapitaOne) ∩ Mosaic using the UPRN as a Unique Identifier which has resulted in 52.3% match which was actually a good match. But there were factors such as duplicate and null values which would not really be useful for matching across the systems and finding the unique detail of a person. The result of this is shown in the ***Primary Attempt*** of the ***Results*** section.

Since Synergy had more than half of UPRN's column empty, more UPRN numbers were fetched from the Address Base file to the Synergy dataset and that newly added column was named as Address_base_UPRN. The address base file consists of the address of the people with their UPRN number which is registered by the company name Ordnance Survey for the Southwark Council. This address_base_UPRN was used in further implementation for the data matching.

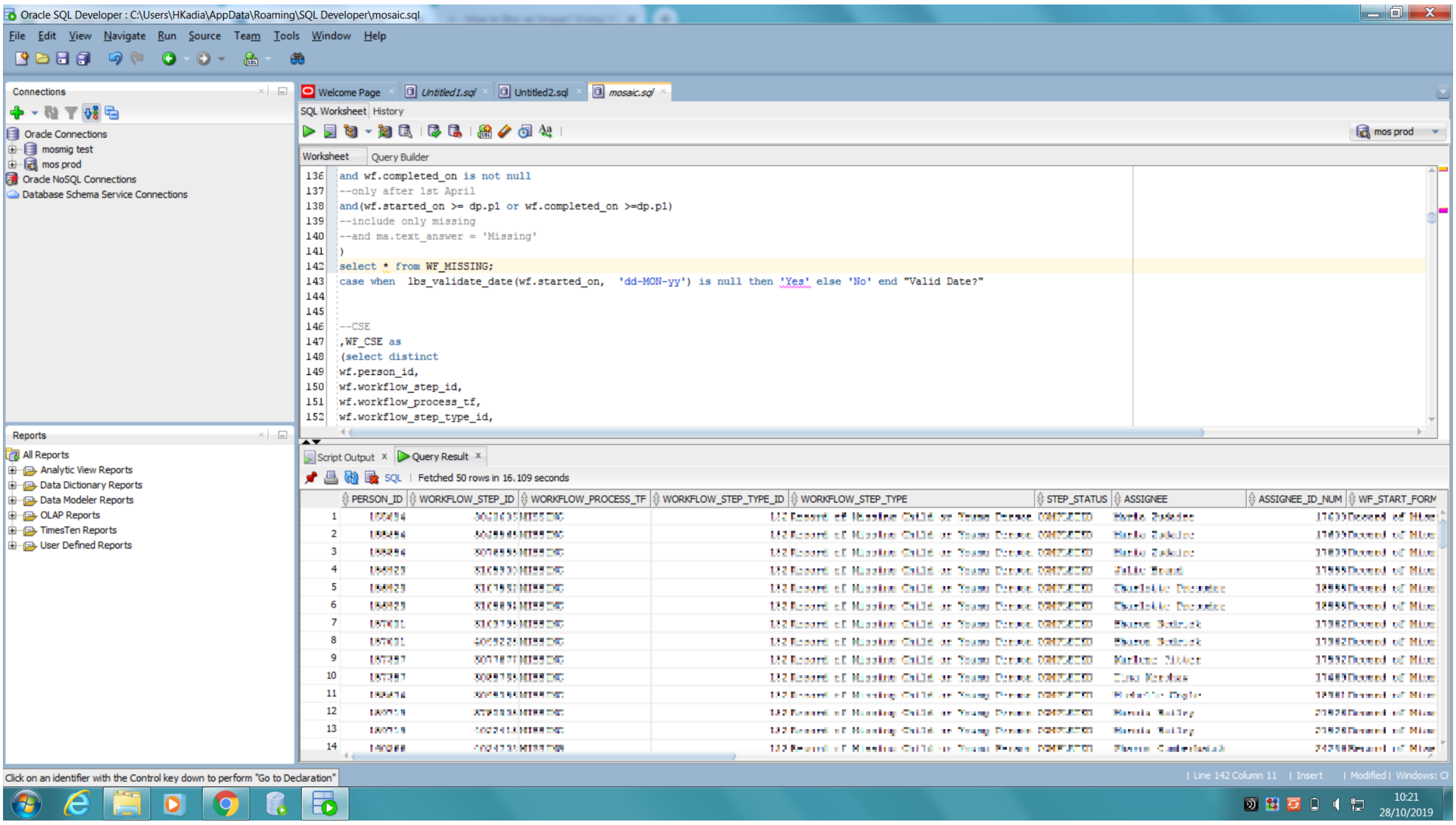
After many trial and error methods of matching across the systems with a unique identifier, UPRN was reliable and so after adding the address base files to the Synergy database, the results changed significantly. The matching result was decreased but on the other hand, this result would be helpful as it was with the unique values and with the added address base file with the Synergy dataset. In this approach I have done with 6 trails for the associative rule using (Synergy ∩ Mosaic) ∩ CapitaOne, (Synergy ∩ CapitaOne) ∩ Mosaic, (Mosaic ∩ Capita) ∩ Synergy, (Mosaic ∩ Synergy) ∩ CapitaOne, (CapitaOne ∩ Synergy) ∩ Mosaic, and (CapitaOne ∩ Mosaic) ∩ Synergy. So, let's say for example we matched Synergy and Mosaic with the UPRN as a unique identifier and stored it into another sheet. After then Mosaic and Synergy were converted to a table and then matched with CapitaOne using the Fuzzy Lookup tool. By this approach based on data quality and uniqueness of the records, the optimum result obtained was 42.3% which is depicted in the ***Final Attempt*** of the ***Results*** section.

The NHS number was present only in the Mosaic database initially. The Council doesn't have any proper unique identifier hence I fetched the NHS numbers form the Mosaic system to Synergy and CapitaOne database based on the following parameters i.e. (First name & Last name) and (First name, last name, date of birth). But this would have a constraint as some of the people might have moved to different housing. But this would have a constraint as some of the people might have moved to different housing.

6. Results

Initially, when I implemented using Oracle SQL Server 19.1.0, the matched records were less as compared to the results of matching in Rstudio and MS Excel due to trimming or hidden white spaces. Hence the results matched were always less than the actual result obtained. The below images depict the result of the matching using Oracle SQL Server and Rstudio code for matching.

Fig. 9 displays the result of the code of matching date of birth, first name, last name, person_id, date of birth, postcode, closure dates, legal status closures and so on. Since I cannot display the data, I have blurred the output of the data.



```
and wf.completed_on is not null
--only after 1st April
and(wf.started_on >= dp.p1 or wf.completed_on >=dp.p1)
--include only missing
--and ma.text_answer = 'Missing'
)
select * from WF_MISSING;
case when  lbs_validate_date(wf.started_on,  'dd-MON-yy') is null then 'Yes' else 'No' end "Valid Date?"


--CSE
,WF_CSE as
(select distinct
wf.person_id,
wf.workflow_step_id,
wf.workflow_process_tf,
wf.workflow_step_type_id,
```

Fetched 50 rows in 16.109 seconds

	PERSON_ID	WORKFLOW_STEP_ID	WORKFLOW_PROCESS_TF	WORKFLOW_STEP_TYPE_ID	WORKFLOW_STEP_TYPE	STEP_STATUS	ASSIGNEE	ASSIGNEE_ID_NUM	WF_START_FORM
1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
12	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
13	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
14	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Click on an identifier with the Control key down to perform "Go to Declaration"

| Line 142 Column 11 | Insert | Modified | Windows: CR

Fig. 9: Mosaic SQL match code output

For the task of cross-checking the results obtained, I cross-checked using Rstudio but the match results were not the same. The results of MS Excel and Rstudio were the same, due to this reason the idea of doing it in the Oracle SQL Server was dropped. The Rstudio code for Cross-checking Mosaic data to Synergy data using the date of birth as a common parameter is attached in ***Appendix 7***.

6.1 Primary Attempt

The table below describes the total values in a dataset and the number of null values in a dataset.

	Mosaic	Synergy	CapitaOne
Values	14399	39876	10624
Null	1985	35118	32921

The table below shows the duplicate and unique values from the dataset of UPRN (used as a Unique Identifier).

	Mosaic	Synergy	CapitaOne
Duplicate	12963	65360	4365
Unique	1368	24952	8242

By the results of integrating various datasets in an associate law. The best result achieved was by (Mosaic ∩ Synergy) ∩ CapitaOne with 4313 values matched in total across three datasets. Since the value 8242 is the least across all those three datasets, the maximum possibility of matching would be 8242 with a 100% match. But here, we get the result of matching as 52.3% (4313/8242).

The results of associative matching by individual means is shown in the table below:

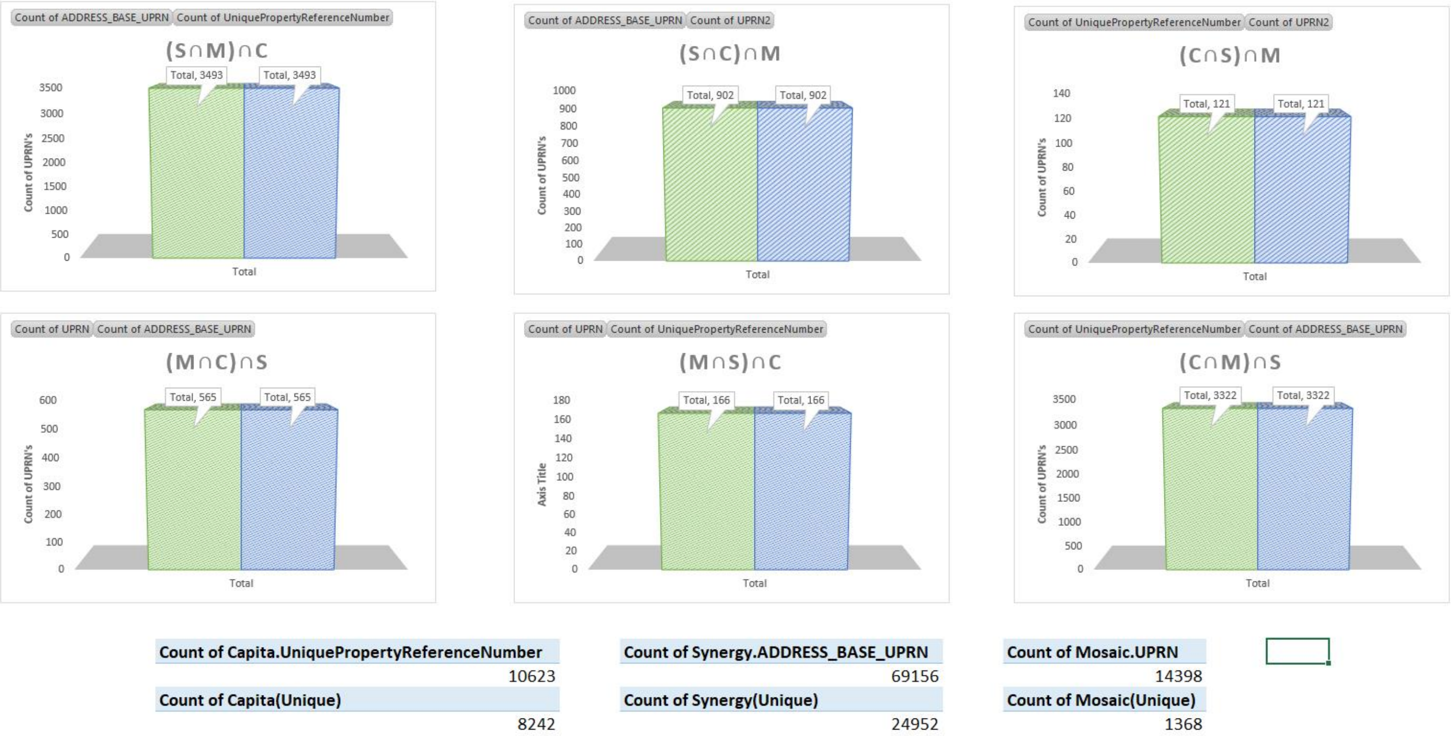
Methods	Results (Records matched 100%)
(Mosaic ∩ Synergy) ∩ CapitaOne	4313
(Mosaic ∩ CapitaOne) ∩ Synergy	140
(Synergy ∩ CapitaOne) ∩ Mosaic	190

6.2 Final Attempt

The optimum result obtained was when the Address_Base_UPRN was used in Synergy which added more UPRN number to the Synergy dataset and based on that the associative rule was applied along with Fuzzy Lookup. The results of each approach are as follows.

Methods	Results (Records matched 100%)
(Synergy ∩ Mosaic) ∩ CapitaOne	3493
(Synergy ∩ CapitaOne) ∩ Mosaic	902
(Mosaic ∩ CapitaOne) ∩ Synergy	565
(Mosaic ∩ Synergy) ∩ CapitaOne	166
(CapitaOne ∩ Synergy) ∩ Mosaic	121
(CapitaOne ∩ Mosaic) ∩ Synergy	3322

The best results obtained by the approach (Synergy ∩ Mosaic) ∩ CapitaOne was 42.3% (3493/8242). The insights of the final attempt are shown in **Fig. 9** given below. The dashboard was created using Pivot tables of each associate law and then all the 6 graphs merged all together in a new sheet.



Count of Capita.UniquePropertyReferenceNumber
10623
Count of Capita(Unique)
8242

Count of Synergy.ADDRESS_BASE_UPRN
69156
Count of Synergy(Unique)
24952

Count of Mosaic.UPRN
14398
Count of Mosaic(Unique)
1368

Fig. 9: Dashboard

6.3 NHS number Fetching

As for the future, Southwark Council's members are working on assigning a unique identifier for each and individual person in their system. The amount of NHS number fetched from Mosaic to the Synergy and CapitaOne are as follows:

Criteria	NHS number (Synergy)
First name, Last name	1002
First name, Last name, Date of Birth	314

Criteria	NHS number (CapitaOne)
First name, Last name	2233
First name, Last name, Date of Birth	773

7. Conclusion

7.1 Potential improvements to the database

There are a number of potential enhancements that can be made to improve the database's usability. One of the main problems faced by this project was the lack of data. Mostly in all of their databases, Missing Data was the biggest concern as the quality of the data was not good which resulted in the percentage of matching accuracy. Hence, there should be more focus on data profiling, data quality, and master data management.

For the improvements in the existing database, the percentage match of over 96% and above can be fixed by correcting it into the original database so that in the near future it would give the optimum result of the data matching. The match between 96% to 99.99% were the problems of manual errors like spelling mistakes while entering into the original database, the formatting of the data column, reducing the number of duplicates as it would result in redundancy of the data, data of birth in the correct format should be generalized among all three systems. There should be validation criteria while entering into the database so that if a person extracts the database into any other format then it should remain the same according to the other database.

Also, not only the Unique Identifiers but also if the data matched with the first name, last name, date of birth, and postcode results in 100% accuracy then the Unique Identifiers should be fetched across the systems so that it will be updated and result in a betterment of the data quality. Moreover, the number of duplicates results in the redundancy of the data. After formatting the records as number format, the number of duplicates of the Unique Identifier considered as UPRN (Unique Property Reference Number) was 12963 for Mosaic, 95360 for Synergy and 4365 for Capita One. Whereas the number of unique values was overpowered by the duplicates as the unique values of UPRN were just reported as 1368 for Mosaic, 24952 for Synergy and 8242 for Capita One. In short, the number of duplicates was more than half in Mosaic and Synergy while the number of unique values of UPRN was twice the number of duplicates.

7.2 Further use of Methodology at the Council

Although there are plenty of tools available for the data integration, Fuzzy Logic in MS Excel was the optimum for this project. The main reason was its adaptability, Fuzzy Logic is a plug-in for MS Excel 2010 and above which everyone can use it even an administrator can use it who has the know-how of MS Excel. Now, since this Fuzzy Logic is pretty handy, Southwark Council has decided to use it in their data to day work and the data matching across two systems is given to a Business Support Officer as an additional work which will indeed save some money rather than outsourcing the data matching project. Now, it may be outsourced only for the Data Quality and the statutory returns of an individual department for the Southwark

Council. One of the managers told me that, no one performed the data matching process in the Southwark Council, and he has seen people manually matching at his previous work and he also added that it was a very time-consuming process and prone to human error. Often the employees spend weeks, trying to cross-check between the system/spreadsheets manually. Along with the majority of end-users being unable to write SQL codes, so would need to be provided with spreadsheets, etc creating a long, and unmanageable process.

An automated process of using Fuzzy Logic tools took some initial research work to be implemented but it would now save time in the long run rather than SQL queries as well. Because Fuzzy Logic works on the same principle of Join operation the same as in SQL but additionally providing with the similarity score as well.

There would be an increased volume in the accuracy, an automated process, once the rules have been established, would take hours if not minutes to provide a complete list of possible matches from the system. There would also be an expected increased higher matching of accuracy as the algorithms should be able to handle misspelled/similar names more objectively than a manual human process. Tasks including the confirmation of the address from the address base files of https://www.ordnancesurvey.co.uk/business-and-government/products/addressbase-products.html have over 40 million addresses and comparing those to Council's collection of manual ones would be a rather time-consuming task.

To sum up, if more than one databases needs to be matched using this approach of Fuzzy Logic in MS Excel then the associate law can be implemented by assuming the databases as A, B and C and can be implemented in many ways as depicted in the Methodology of the project (A ∩ B) ∩ C, A ∩ (B ∩ C) and (B ∩ C) ∩ A, B ∩ (C ∩ A), C ∩ (B ∩ A) and (C ∩ B) ∩ A.

8. Further Work

On behalf of taxpayers, many of them will argue that they have already spent a lot on these families, why the government spending more and the local government is determined to get the results and value for money that will save money for the taxpayer in the end. Those are the difficult challenges for any government but this immense task will take new ways of thinking and will take committed local actions, flexibility and huge perseverance. People in Troubled Families aren't worthless nor are they programmed to fail and in order to help them and turn their lives around and heal the scars of the broken society, every local government is now planning a data matching across their departments and get an insights into their expenditure and the number of people having an advantage of this programme and also those who aren't able to get an advantage of the benefits which they should get.

The work I have done is just across three systems of the Council that would further be input into their Management Information System by their authorized persons. There are many more departments on which this would be implemented into their Management Information System. Also, when defining a Unique Identifier and improvising the data quality would come into play to an extent so that services such as NEET (Not in Education or Training), SEND (Special Education Needs Disability) and EHC (Educational Health Care Plan) which are not currently in process will bring an ignition to those services for evaluating the children between 16-18 who are not in University or College nor in an employment.

9. References

An Overview of Fuzzy Name Matching Techniques, 2018. Available from: https://www.rosette.com/blog/overview-fuzzy-name-matching-techniques/ .

CAI, X.Y., Wei, S.H.E.N., LU, L.N., ZHU, C.W., ZENG, C.Z. and MENG, X.Y., 2017. The Application of Case Teaching in Excel Lookup and Reference Functions. DEStech Transactions on Social Science, Education and Human Science, (meit) Available from http://dpi-proceedings.com/index.php/dtssehs/article/view/12826

Daniel J. Steinbock (2005). DATA MATCHING, DATA MINING, AND DUE PROCESS. Georgia Law Review. 40 1-1245. https://heinonline.org/HOL/Page?handle=hein.journals/geolr40&div=8&g_sent=1&casa_token=&collection=journals .

Data Discovery &amp; Profiling, 2019. Available from: https://www.ataccama.com/product/data-discovery-and-profiling .

De Mantaras, R.L. and Plaza, E., 1997. Case-Based Reasoning: an overview. AI communications, 10(1), pp.21-29.

Doan, A., Lu, Y., Lee, Y. and Han, J., 2003. Profile-based object matching for information integration. IEEE Intelligent Systems, 2003 Sep; 18(5), pp.54-59. Available from https://ieeexplore.ieee.org/document/1234770

Dusetzina, S.B., Tyree, S., Meyer, A.M., Meyer, A., Green, L. and Carpenter, W.R., 2014. Linking data for health services research: a framework and instructional guide. Available from https://www.ncbi.nlm.nih.gov/books/NBK253312/

Full-Text Search Functions, 2019. Available from: https://dev.mysql.com/doc/refman/5.7/en/fulltext-search.html .

Fuzzy Matching Algorithms To Help Data Scientists Match Similar Data, Jan 2016. Available from: https://www.datasciencecentral.com/profiles/blogs/fuzzy-matching-algorithms-to-help-data-scientists-match-similar

Gangemi, A. (2015). Pushing the Limits of Instance Matching Systems, Republic and Canton of Geneva: International World Wide Web Conferences Steering Committee.

H. Shimazu, H. Kitano, and A. Shibata. Retrieving cases from relational databases: another strike toward corporatewide case-based systems. In Proc. 13th Intern. Joint Conference on Artificial Intelligence (IJCAI'93), pages 909–914, 1993.

Hammill, B.G., Hernandez, A.F., Peterson, E.D., Fonarow, G.C., Schulman, K.A. and Curtis, L.H., 2009. Linking inpatient clinical registry data to Medicare claims data using indirect identifiers. American heart journal, 157(6), pp.995-1000.

Kao, A., Poteet, S.R., Poteet, S.R. (2007). Natural Language Processing and Text Mining London: Springer.

Li Rong. Skillfully using Vlookup Function to Complete Information Check—Application in the Check of Students Admission Information. Computer Development & Applications, 2015(1):60-62

Li, Q., Glynn, R.J., Dreyer, N.A., Liu, J., Mogun, H. and Setoguchi, S., 2011. Validity of claims-based definitions of left ventricular systolic dysfunction in Medicare patients. Pharmacoepidemiology and drug safety, 20(7), pp.700-708.

M. Bilenko and R. Mooney, 2002, Learning to Combine Trained Distance Metrics for Duplicate Detection in Databases, tech. report AI 02- 296.

Marsolo, K., 2012. Approaches to facilitate institutional review board approval of multicenter research studies. Medical care, pp.S77-S81.

Megter (2016). Fuzzy Matching Algorithms To Help Data Scientists Match Similar Data. Fuzzy Matching Algorithms To Help Data Scientists Match Similar Data. Available from: https://www.datasciencecentral.com/profiles/blogs/fuzzy-matching-algorithms-to-help-data-scientists-match-similar .

Ministry of Housing, and Communities and Local GovernmentNational evaluation of the Troubled Families Programme 2015-2020: Findings. National evaluation of the Troubled Families Programme 2015-2020: Findings. Available from: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/786889/National_evaluation_of_the_Troubled_Families_Programme_2015_to_2020_evaluation_overview_policy_report.pdf

Multi-agency threshold guide, Southwark Safeguarding Children Board, 2019.

Portinale, L. and Montani, S., 2008, November. A fuzzy logic approach to case matching and retrieval suitable to SQL implementation. In 2008 20th IEEE International Conference on Tools with Artificial Intelligence (Vol. 2, pp. 241-245). IEEE.

Reviews for Data Quality Tools, 2019. Available from: https://www.gartner.com/reviews/market/data-quality-tools .

Richard Baxter (2017). No Exact Match? How to Match Similar Data Tables in Excel with Fuzzy Lookup.

S. Sarawagi and A. Bhamidipaty, 2002, “Interactive Deduplication Using Active Learning,” Proc. 8th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (SIGKDD 02).

S. Tejada, C. Knoblock, and S. Minton, 2002, “Learning Domain-Independent String **Transformation** Weights for High Accuracy Object Identification,” Proc. 8th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (SIGKDD 02).

Saveta, T., Daskalaki, E., Flouris, G., Fundulaki, I., Herschel, M., Ngonga Ngomo, A. (May 18, 2015). Pushing the Limits of Instance Matching Systems. ACM, 105-106.

Schumacher, J. and Bergmann, R., 2000, September. An efficient approach to similarity-based retrieval on top of relational databases. In European Workshop on Advances in Case-Based Reasoning (pp. 273-285). Springer, Berlin, Heidelberg.

Southwark Council (2018). Troubled Families Outcomes Plan Phase 2, Southwark Council

Southwark Families Matter Strategy&nbsp, 2018; Available from https://www.southwark.gov.uk/assets/attach/4964/Southwark-Families-Matter-Strategy.pdf .

Susan Li (2019). Natural Language Processing for Fuzzy String Matching with Python. Natural Language Processing for Fuzzy String Matching with Python. Available from: https://towardsdatascience.com/natural-language-processing-for-fuzzy-string-matching-with-python-6632b7824c49 .

Sveshnikov, S. and Bocharnikov, V., 2010. Fuzzy for Excel, User Manual. User Manual (June 10, 2010).

Ul Hassan, Z., Naeem, M., Khalid, M. (2015). Proposed Generic Full Text Searching Algorithm: A Database Approach. International Journal of Computer & Organization Trends. 22 (1), 14-15. Available from https://www.researchgate.net/publication/282353316_Proposed_Generic_Full_Text_Searching_Algorithm_A_Database_Approach.

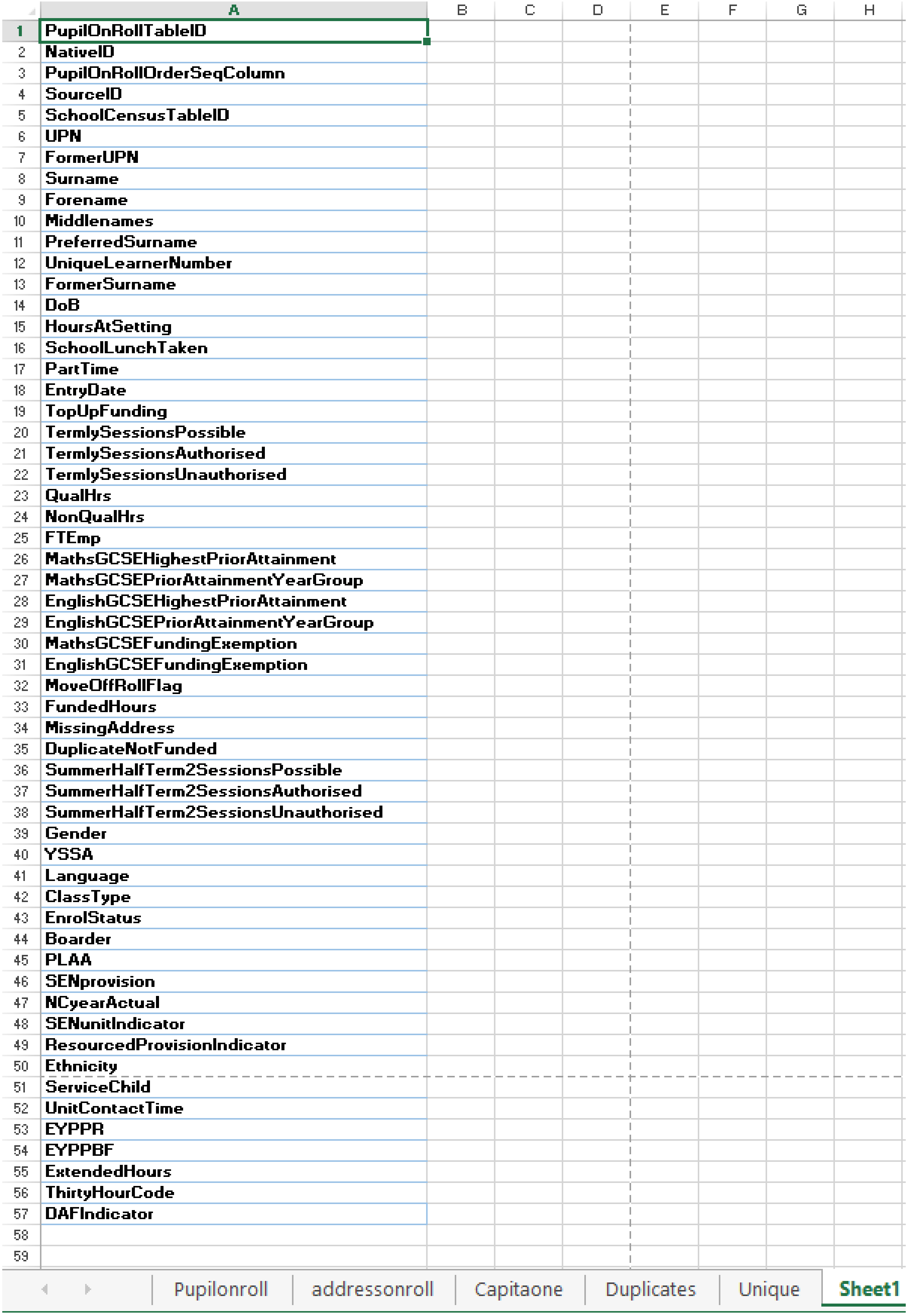
Wang, S. and Jiang, J. (2016). A Compare-Aggregate Model for Matching Text Sequences. Available from https://arxiv.org/abs/1611.01747 .

What is local government?, 2019, Available from https://www.local.gov.uk/about/what-local-government .

Winglee, M., Valliant, R. and Scheuren, F., 2005. A case study in record linkage. Survey Methodology, 31(1), pp.3-11.

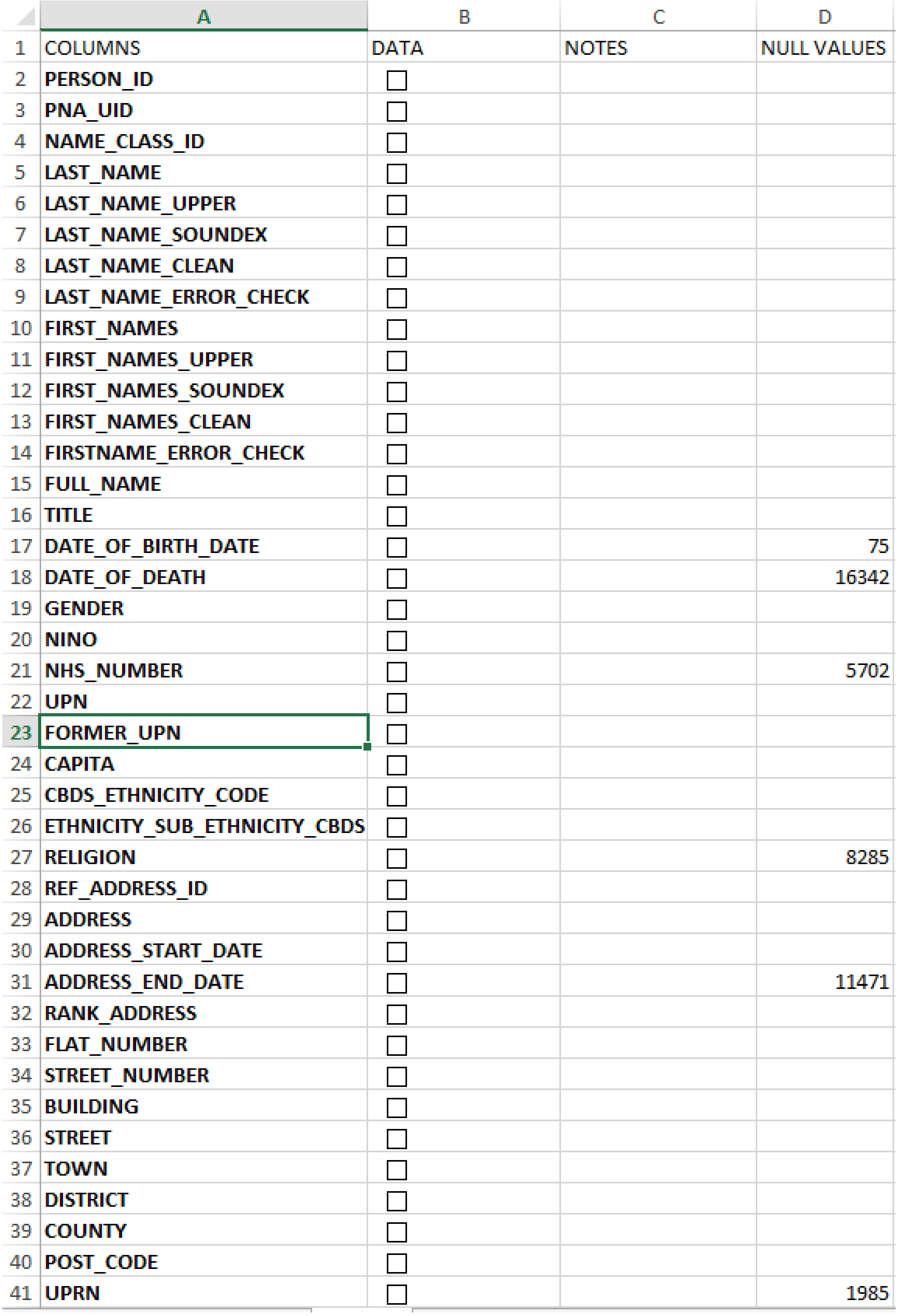
A. Appendix

1. Capita One (School Census)

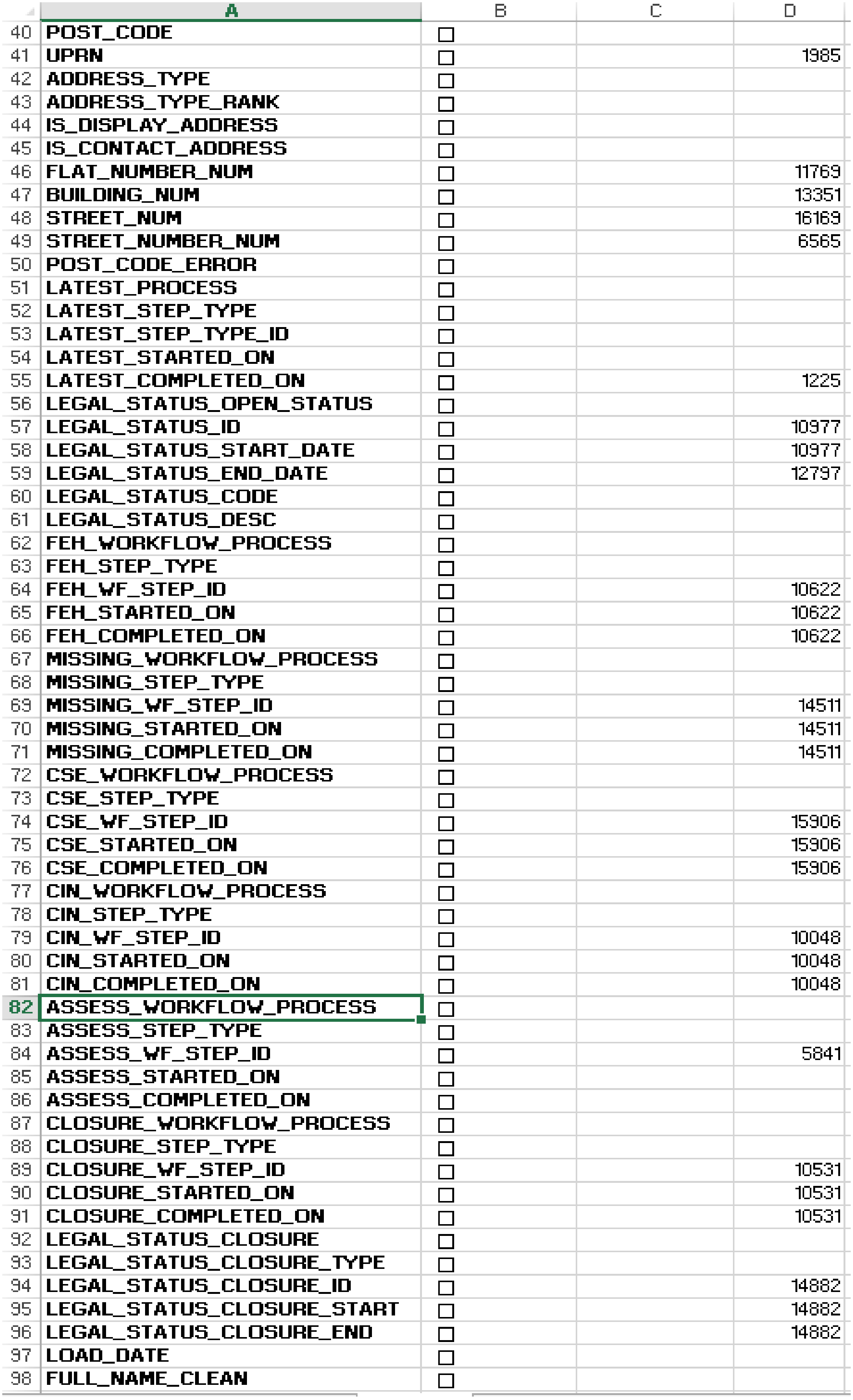


	A	B	C	D	E	F	G	H
1	PupilOnRollTableID							
2	NativeID							
3	PupilOnRollOrderSeqColumn							
4	SourceID							
5	SchoolCensusTableID							
6	UPN							
7	FormerUPN							
8	Surname							
9	Forename							
10	Middlenames							
11	PreferredSurname							
12	UniqueLearnerNumber							
13	FormerSurname							
14	DoB							
15	HoursAtSetting							
16	SchoolLunchTaken							
17	PartTime							
18	EntryDate							
19	TopUpFunding							
20	TermlySessionsPossible							
21	TermlySessionsAuthorised							
22	TermlySessionsUnauthorised							
23	QualHrs							
24	NonQualHrs							
25	FTEmp							
26	MathsGCSEHighestPriorAttainment							
27	MathsGCSEPriorAttainmentYearGroup							
28	EnglishGCSEHighestPriorAttainment							
29	EnglishGCSEPriorAttainmentYearGroup							
30	MathsGCSEFundingExemption							
31	EnglishGCSEFundingExemption							
32	MoveOffRollFlag							
33	FundedHours							
34	MissingAddress							
35	DuplicateNotFunded							
36	SummerHalfTerm2SessionsPossible							
37	SummerHalfTerm2SessionsAuthorised							
38	SummerHalfTerm2SessionsUnauthorised							
39	Gender							
40	YSSA							
41	Language							
42	ClassType							
43	EnrolStatus							
44	Boarder							
45	PLAA							
46	SENprovision							
47	NCyearActual							
48	SENunitIndicator							
49	ResourcedProvisionIndicator							
50	Ethnicity							
51	ServiceChild							
52	UnitContactTime							
53	EYPPR							
54	EYPPBF							
55	ExtendedHours							
56	ThirtyHourCode							
57	DAFIndicator							
58								
59								

2. Mosaic (Children Social Care)

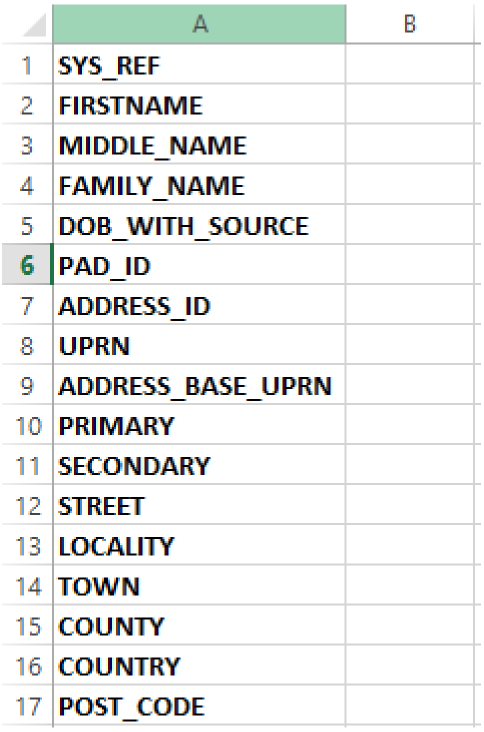


	A	B	C	D
1	COLUMNS	DATA	NOTES	NULL VALUES
2	PERSON_ID	□		
3	PNA_UID	□		
4	NAME_CLASS_ID	□		
5	LAST_NAME	□		
6	LAST_NAME_UPPER	□		
7	LAST_NAME_SOUNDEX	□		
8	LAST_NAME_CLEAN	□		
9	LAST_NAME_ERROR_CHECK	□		
10	FIRST_NAMES	□		
11	FIRST_NAMES_UPPER	□		
12	FIRST_NAMES_SOUNDEX	□		
13	FIRST_NAMES_CLEAN	□		
14	FIRSTNAME_ERROR_CHECK	□		
15	FULL_NAME	□		
16	TITLE	□		
17	DATE_OF_BIRTH_DATE	□		75
18	DATE_OF_DEATH	□		16342
19	GENDER	□		
20	NINO	□		
21	NHS_NUMBER	□		5702
22	UPN	□		
23	FORMER_UPN	□		
24	CAPITA	□		
25	CBDS_ETHNICITY_CODE	□		
26	ETHNICITY_SUB_ETHNICITY_CBDS	□		
27	RELIGION	□		8285
28	REF_ADDRESS_ID	□		
29	ADDRESS	□		
30	ADDRESS_START_DATE	□		
31	ADDRESS_END_DATE	□		11471
32	RANK_ADDRESS	□		
33	FLAT_NUMBER	□		
34	STREET_NUMBER	□		
35	BUILDING	□		
36	STREET	□		
37	TOWN	□		
38	DISTRICT	□		
39	COUNTY	□		
40	POST_CODE	□		
41	UPRN	□		1985



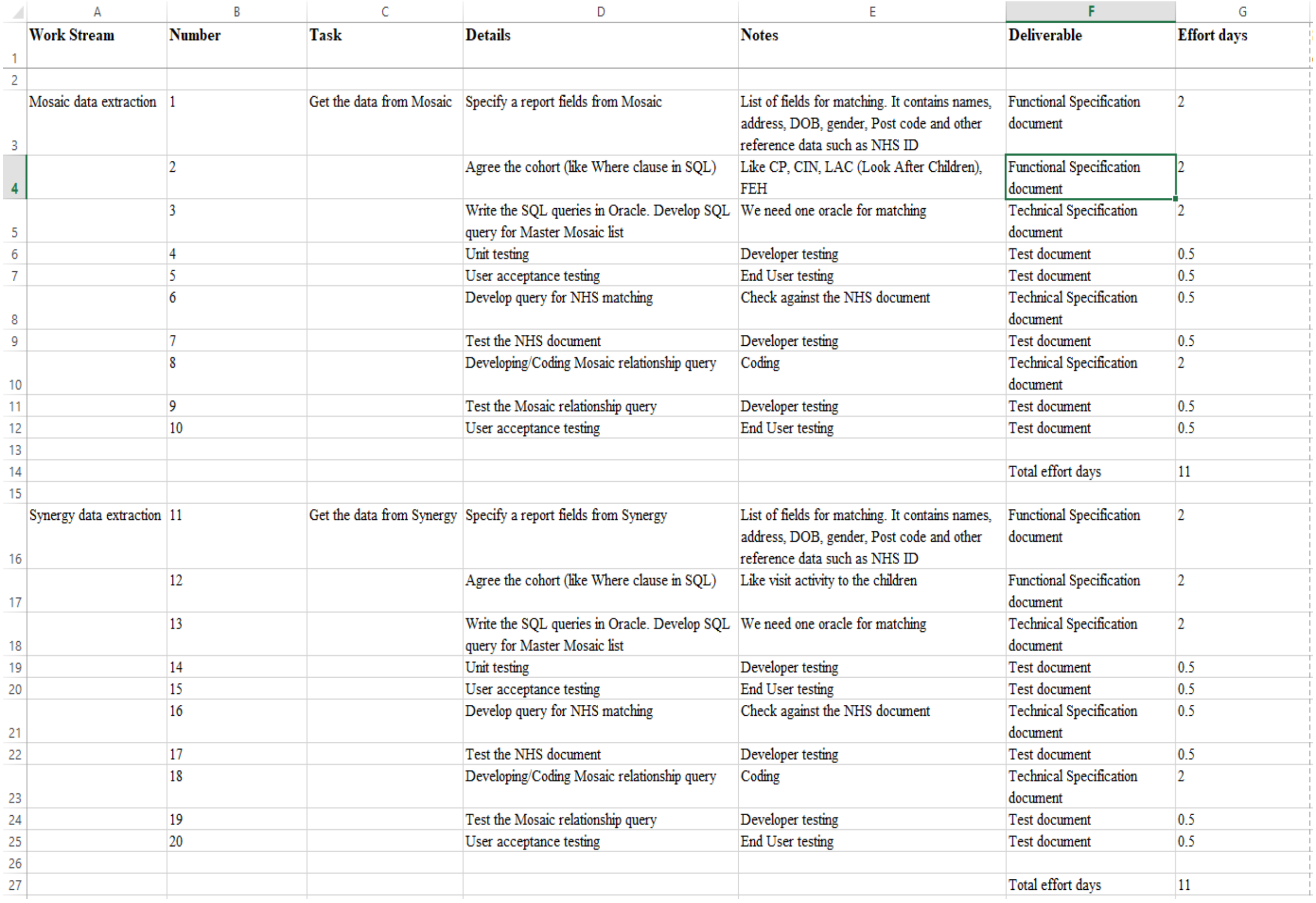
	A	B	C	D
40	POST_CODE	□		
41	UPRN	□		1985
42	ADDRESS_TYPE	□		
43	ADDRESS_TYPE_RANK	□		
44	IS_DISPLAY_ADDRESS	□		
45	IS_CONTACT_ADDRESS	□		
46	FLAT_NUMBER_NUM	□		11769
47	BUILDING_NUM	□		13351
48	STREET_NUM	□		16169
49	STREET_NUMBER_NUM	□		6565
50	POST_CODE_ERROR	□		
51	LATEST_PROCESS	□		
52	LATEST_STEP_TYPE	□		
53	LATEST_STEP_TYPE_ID	□		
54	LATEST_STARTED_ON	□		
55	LATEST_COMPLETED_ON	□		1225
56	LEGAL_STATUS_OPEN_STATUS	□		
57	LEGAL_STATUS_ID	□		10977
58	LEGAL_STATUS_START_DATE	□		10977
59	LEGAL_STATUS_END_DATE	□		12797
60	LEGAL_STATUS_CODE	□		
61	LEGAL_STATUS_DESC	□		
62	FEH_WORKFLOW_PROCESS	□		
63	FEH_STEP_TYPE	□		
64	FEH_WF_STEP_ID	□		10622
65	FEH_STARTED_ON	□		10622
66	FEH_COMPLETED_ON	□		10622
67	MISSING_WORKFLOW_PROCESS	□		
68	MISSING_STEP_TYPE	□		
69	MISSING_WF_STEP_ID	□		14511
70	MISSING_STARTED_ON	□		14511
71	MISSING_COMPLETED_ON	□		14511
72	CSE_WORKFLOW_PROCESS	□		
73	CSE_STEP_TYPE	□		
74	CSE_WF_STEP_ID	□		15906
75	CSE_STARTED_ON	□		15906
76	CSE_COMPLETED_ON	□		15906
77	CIN_WORKFLOW_PROCESS	□		
78	CIN_STEP_TYPE	□		
79	CIN_WF_STEP_ID	□		10048
80	CIN_STARTED_ON	□		10048
81	CIN_COMPLETED_ON	□		10048
82	ASSESS_WORKFLOW_PROCESS	□		
83	ASSESS_STEP_TYPE	□		
84	ASSESS_WF_STEP_ID	□		5841
85	ASSESS_STARTED_ON	□		
86	ASSESS_COMPLETED_ON	□		
87	CLOSURE_WORKFLOW_PROCESS	□		
88	CLOSURE_STEP_TYPE	□		
89	CLOSURE_WF_STEP_ID	□		10531
90	CLOSURE_STARTED_ON	□		10531
91	CLOSURE_COMPLETED_ON	□		10531
92	LEGAL_STATUS_CLOSURE	□		
93	LEGAL_STATUS_CLOSURE_TYPE	□		
94	LEGAL_STATUS_CLOSURE_ID	□		14882
95	LEGAL_STATUS_CLOSURE_START	□		14882
96	LEGAL_STATUS_CLOSURE_END	□		14882
97	LOAD_DATE	□		
98	FULL_NAME_CLEAN	□		

3. Synergy (Children Centres)

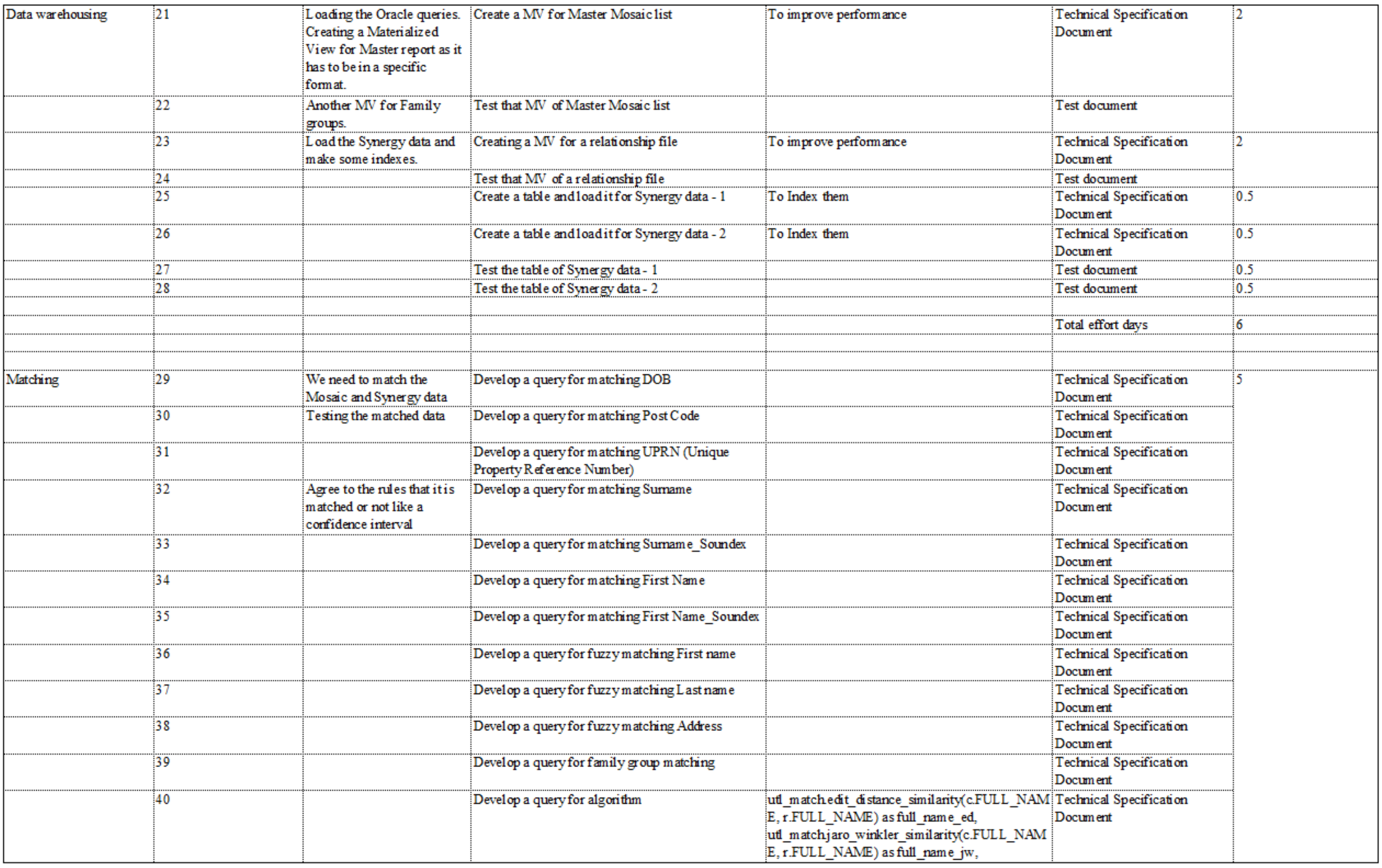


	A	B
1	SYS_REF	
2	FIRSTNAME	
3	MIDDLE_NAME	
4	FAMILY_NAME	
5	DOB_WITH_SOURCE	
6	PAD_ID	
7	ADDRESS_ID	
8	UPRN	
9	ADDRESS_BASE_UPRN	
10	PRIMARY	
11	SECONDARY	
12	STREET	
13	LOCALITY	
14	TOWN	
15	COUNTY	
16	COUNTRY	
17	POST_CODE	

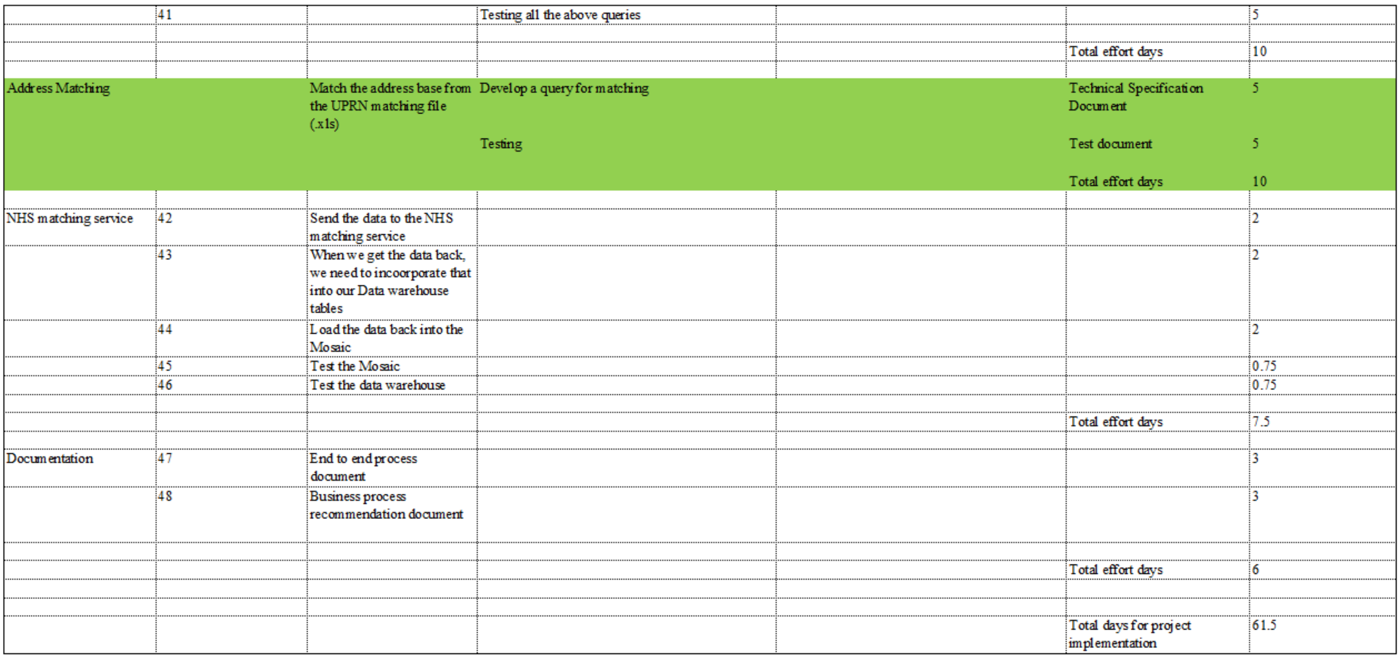
4. Project Pre-Plan



	A	B	C	D	E	F	G
1	**Work Stream**	**Number**	**Task**	**Details**	**Notes**	**Deliverable**	**Effort days**
2							
3	Mosaic data extraction	1	Get the data from Mosaic	Specify a report fields from Mosaic	List of fields for matching. It contains names, address, DOB, gender, Post code and other reference data such as NHS ID	Functional Specification document	2
4		2		Agree the cohort (like Where clause in SQL)	Like CP, CIN, LAC (Look After Children), FEH	Functional Specification document	2
5		3		Write the SQL queries in Oracle. Develop SQL query for Master Mosaic list	We need one oracle for matching	Technical Specification document	2
6		4		Unit testing	Developer testing	Test document	0.5
7		5		User acceptance testing	End User testing	Test document	0.5
8		6		Develop query for NHS matching	Check against the NHS document	Technical Specification document	0.5
9		7		Test the NHS document	Developer testing	Test document	0.5
10		8		Developing/Coding Mosaic relationship query	Coding	Technical Specification document	2
11		9		Test the Mosaic relationship query	Developer testing	Test document	0.5
12		10		User acceptance testing	End User testing	Test document	0.5
13							
14						Total effort days	11
15							
16	Synergy data extraction	11	Get the data from Synergy	Specify a report fields from Synergy	List of fields for matching. It contains names, address, DOB, gender, Post code and other reference data such as NHS ID	Functional Specification document	2
17		12		Agree the cohort (like Where clause in SQL)	Like visit activity to the children	Functional Specification document	2
18		13		Write the SQL queries in Oracle. Develop SQL query for Master Mosaic list	We need one oracle for matching	Technical Specification document	2
19		14		Unit testing	Developer testing	Test document	0.5
20		15		User acceptance testing	End User testing	Test document	0.5
21		16		Develop query for NHS matching	Check against the NHS document	Technical Specification document	0.5
22		17		Test the NHS document	Developer testing	Test document	0.5
23		18		Developing/Coding Mosaic relationship query	Coding	Technical Specification document	2
24		19		Test the Mosaic relationship query	Developer testing	Test document	0.5
25		20		User acceptance testing	End User testing	Test document	0.5
26							
27						Total effort days	11



Data warehousing	21	Loading the Oracle queries. Creating a Materialized View for Master report as it has to be in a specific format.	Create a MV for Master Mosaic list	To improve performance	Technical Specification Document	2
	22	Another MV for Family groups.	Test that MV of Master Mosaic list		Test document	
	23	Load the Synergy data and make some indexes.	Creating a MV for a relationship file	To improve performance	Technical Specification Document	2
	24		Test that MV of a relationship file		Test document	
	25		Create a table and load it for Synergy data - 1	To Index them	Technical Specification Document	0.5
	26		Create a table and load it for Synergy data - 2	To Index them	Technical Specification Document	0.5
	27		Test the table of Synergy data - 1		Test document	0.5
	28		Test the table of Synergy data - 2		Test document	0.5
					Total effort days	6
Matching	29	We need to match the Mosaic and Synergy data	Develop a query for matching DOB		Technical Specification Document	5
	30	Testing the matched data	Develop a query for matching Post Code		Technical Specification Document	
	31		Develop a query for matching UPRN (Unique Property Reference Number)		Technical Specification Document	
	32	Agree to the rules that it is matched or not like a confidence interval	Develop a query for matching Surname		Technical Specification Document	
	33		Develop a query for matching Surname_Soundex		Technical Specification Document	
	34		Develop a query for matching First Name		Technical Specification Document	
	35		Develop a query for matching First Name_Soundex		Technical Specification Document	
	36		Develop a query for fuzzy matching First name		Technical Specification Document	
	37		Develop a query for fuzzy matching Last name		Technical Specification Document	
	38		Develop a query for fuzzy matching Address		Technical Specification Document	
	39		Develop a query for family group matching		Technical Specification Document	
	40		Develop a query for algorithm	utl_match.edit_distance_similarity(c.FULL_NAME, r.FULL_NAME) as full_name_ed, utl_match.jaro_winkler_similarity(c.FULL_NAME, r.FULL_NAME) as full_name_jw,	Technical Specification Document	



	41		Testing all the above queries			5
					Total effort days	10
Address Matching		Match the address base from the UPRN matching file (.xls)	Develop a query for matching		Technical Specification Document	5
			Testing		Test document	5
					Total effort days	10
NHS matching service	42	Send the data to the NHS matching service				2
	43	When we get the data back, we need to incoorporate that into our Data warehouse tables				2
	44	Load the data back into the Mosaic				2
	45	Test the Mosaic				0.75
	46	Test the data warehouse				0.75
					Total effort days	7.5
Documentation	47	End to end process document				3
	48	Business process recommendation document				3
					Total effort days	6
					Total days for project implementation	61.5

5. Mosaic SQL Code

```
--CREATE OR REPLACE FORCE VIEW "FW"."LBS_V_CSC_TF_RISK_DQ" as

  WITH date_Para as
(select
--to_date('&P1&','dd/mm/yyyy') P1,
to_date('01/04/2015','dd/mm/yyyy') P1
--to_date('&P2&','dd/mm/yyyy') P2
--to_date('23/11/2018','dd/mm/yyyy') P2
from dual)
--select * from date_Para;

--LEGAL STATUS
,LEGAL_STATUS AS
 (select distinct
la.PERSON_ID,
la.LEGAL_STATUS_ID,
la.LEGAL_STATUS_CODE,
la.LEGAL_STATUS_DESC,
la.legal_status_ranked_desc,
la.LEGAL_STATUS_START_DATE,
la.legal_status_open_status,
la.LEGAL_STATUS_END_DATE,
nvl(la.LEGAL_STATUS_END_DATE,to_date('01/01/1900','dd/mm/yyyy'))
LEGAL_STATUS_END_DATE_no_null,
DENSE_RANK () OVER (PARTITION by la.PERSON_ID ORDER BY
la.LEGAL_STATUS_START_DATE asc, LA.LEGAL_STATUS_END_DATE asc, la.LEGAL_STATUS_ID
asc) RANK_ASC,
DENSE_RANK () OVER (PARTITION by la.PERSON_ID ORDER BY
la.LEGAL_STATUS_START_DATE desc, LA.LEGAL_STATUS_END_DATE desc,
la.LEGAL_STATUS_ID desc) RANK_DESC
from lbs_mv_csc_lac la
cross join date_Para dp
where la.legal_status_check  = 'Include'
and (la.legal_status_end_date is null or la.legal_status_end_date >= dp.p1)
)

--select LS.*
--,case when  lbs_validate_date(LS.legal_status_end_date,  'dd-MON-yy') is null then 'NO'
else 'YES' end "Valid Date?"
--case when  lbs_validate_date(LS.legal_status_start_date,  'dd-MON-yy') is null then 'Yes'
else 'No' end "Valid Date?"
--from LEGAL_STATUS LS;

,WF_FEH as
(select distinct
wf.person_id,
wf.workflow_step_id,
wf.workflow_process_tf,
wf.workflow_step_type_id,
wf.workflow_step_type,
wf.step_status,
wf.assignee,
wf.assignee_id_num,
wf.wf_start_form_field,
wf.wf_start_form_id,
wf.started_on,
wf.wf_end_form_field,
wf.wf_end_form_id,
coalesce(wf.wf_start_form_id,wf.wf_end_form_id) form_id_start_end,
wf.completed_on,

RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON asc,
wf.COMPLETED_ON asc,wf.WORKFLOW_STEP_ID asc) RANK_ASC,
RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON desc,
wf.COMPLETED_ON desc,wf.WORKFLOW_STEP_ID desc) RANK_DESC
from lbs_mv_asc_workflow_ranking wf
cross join date_para dp

where
wf.person_id <> 500000
--troubled families workflow
and wf.workflow_process_tf ='FEH'
--393     zzzEarly Help - Case Closure
--2944   Family Early Help Internal Social Care Referral
--3008   SFFT Delivery Plan
--3009   SFFT Review
--3166   Early Help Closure without File Retention
--3168   Early Help Delivery Plan
--3169   Early Help Delivery Plan Review
--3170   Early Help Introduction Letter
--3173   Early Help Closure with File Retention
--3210   Early Help Review Meeting
--3325   SFFT Migration Step
--3364   SFFT Closure with File Retention
--3365   SFFT without File Retention
--4646   Family Early Help Parenting Group
--4788   SFFT Initial Delivery Plan
--4806   SFFT Review and Plan
--5231   Family Early Help Closure
--5232   Family Early Help Review and Plan
--5233   Family Early Help Assessment and Initial Plan
--5234   Family Early Help Referral
--remove cancelled
and wf.step_status_rank <> 6
--include only those with an end date
and wf.completed_on is not null
--only after 1st April
--and(wf.started_on >= to_date('01/04/2015','dd/mm/yyyy') or wf.completed_on
>=to_date('01/04/2015','dd/mm/yyyy'))
and(wf.started_on >= dp.p1 or wf.completed_on >=dp.p1)
```

```
)

--select * from WF_FEH
--order by PERSON_ID, WF_RANK_ASC;


--Missing
,WF_MISSING as
(select distinct
wf.person_id,
wf.workflow_step_id,
wf.workflow_process_tf,
wf.workflow_step_type_id,
wf.workflow_step_type,
wf.step_status,
wf.assignee,
wf.assignee_id_num,

wf.wf_start_form_field,
wf.wf_start_form_id,

wf.started_on,

wf.wf_end_form_field,
wf.wf_end_form_id,

coalesce(wf.wf_start_form_id,wf.wf_end_form_id) form_id_start_end,

wf.completed_on,

RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON asc,
wf.COMPLETED_ON asc,wf.WORKFLOW_STEP_ID asc) RANK_ASC,
RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON desc,
wf.COMPLETED_ON desc,wf.WORKFLOW_STEP_ID desc) RANK_DESC

from lbs_mv_asc_workflow_ranking wf
cross join date_para dp
inner join (select workflow_step_id, form_id, template_id, mapping_id, text_answer from dm_cached_form_answers
where template_id = 101 and mapping_id = 'FORM1154707360437->1a1ac263-faaf-4861-9152-60f4d677e904' and text_answer = 'Missing' ) ma on wf.workflow_step_id = ma.workflow_step_id
where
wf.person_id <> 500000
--troubled families workflow
and wf.workflow_process_tf ='MISSING'
--remove cancelled
and wf.step_status_rank <> 6
--include only those with an end date
and wf.completed_on is not null
--only after 1st April
```

```
and(wf.started_on >= dp.p1 or wf.completed_on >=dp.p1)
--include only missing
--and ma.text_answer = 'Missing'
)
--select * from WF_MISSING;
--case when  lbs_validate_date(wf.started_on,  'dd-MON-yy') is null then 'Yes' else 'No' end
"Valid Date?"


--CSE
,WF_CSE as
(select distinct
wf.person_id,
wf.workflow_step_id,
wf.workflow_process_tf,
wf.workflow_step_type_id,
wf.workflow_step_type,
wf.step_status,
wf.assignee,
wf.assignee_id_num,

wf.started_on,
wf.wf_end_form_field,
wf.wf_end_form_id,

coalesce(wf.wf_start_form_id,wf.wf_end_form_id) form_id_start_end,
wf.completed_on,
CASE WHEN wf.started_on_form is not null then wf.started_on_form
else wf.completed_on_system end as completed_on_new,
case when mase.workflow_id is not null then 'MASE REVIEW'
else null end as MASE_REVIEW,

RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON asc,
wf.COMPLETED_ON asc,wf.WORKFLOW_STEP_ID asc) RANK_ASC,
RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON desc,
wf.COMPLETED_ON desc,wf.WORKFLOW_STEP_ID desc) RANK_DESC

from lbs_mv_asc_workflow_ranking wf
cross join date_para dp

--has completed case risk assessment
--has completed mase review
inner join (select distinct person_id, workflow_id from lbs_mv_asc_workflow_ranking where
step_status_rank = 5 and workflow_step_type_id = 3604)mase on
wf.workflow_id = mase.workflow_id

where
wf.person_id <> 500000
--troubled families workflow
and wf.workflow_process_tf ='CSE'
--17       CSE Risk Screening Assessment
```

```
--3604   MASE Review
--remove cancelled
and wf.step_status_rank <> 6
--include only those with an end date
and wf.completed_on is not null
--only after 1st April
and(wf.started_on >= dp.p1 or wf.completed_on >=dp.p1)
)

--select * from WF_CSE;

--CIN
,WF_CIN as
(select distinct
wf.person_id,
wf.workflow_step_id,
wf.workflow_process_tf,
wf.workflow_step_type_id,
wf.workflow_step_type,
wf.step_status,
wf.assignee,
wf.assignee_id_num,
wf.wf_start_form_field,
wf.wf_start_form_id,
wf.started_on,
wf.wf_end_form_field,
wf.wf_end_form_id,
coalesce(wf.wf_start_form_id,wf.wf_end_form_id) form_id_start_end,
wf.completed_on,
substr(ma.text_answer,13,7) Plan_completed_time,
to_date(substr(ma.text_answer,24,10),'dd/mm/yyyy') Plan_completed_date,
ma.text_answer Plan_completed_by,

RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON asc,
wf.COMPLETED_ON asc,wf.WORKFLOW_STEP_ID asc) RANK_ASC,
RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON desc,
wf.COMPLETED_ON desc,wf.WORKFLOW_STEP_ID desc) RANK_DESC

from lbs_mv_asc_workflow_ranking wf
cross join date_para dp

left outer join (select workflow_step_id, form_id, template_id, mapping_id, text_answer
from dm_cached_form_answers
where template_id in (254,569)  and mapping_id in ('ICSCINPLAN->3D6E418B-838E-E100-F9A2-B7FF45070C72','REVCONFOUTLINEPLN->3D6E418B-838E-E100-F9A2-B7FF45070C72'))
ma on wf.workflow_step_id = ma.workflow_step_id

where
wf.person_id <> 500000
--troubled families workflow
and wf.workflow_process_tf ='CIN'
```

```
--484     Child Protection Plan
--2029   MIGRATED CIN Plan Review / Network Meeting
--2118   MIGRATED Starting to be Looked After
--4267   CIN Network Review / Updated CIN plan
--4286   Initial CIN Plan
--Child or Young Person in Need Plan
--Child or Young Person in Need Review
--Record of Review Child Protection Conference
--remove cancelled
and wf.step_status_rank <> 6
--include only those with an end date
and wf.completed_on is not null
--only after 1st April
and(wf.started_on >= dp.p1 or wf.completed_on >=dp.p1)
)
--select * from WF_CIN;

--Assessments
,WF_ASSESS as
(select distinct
wf.person_id,
wf.workflow_step_id,
wf.workflow_process_tf,
wf.workflow_step_type_id,
wf.workflow_step_type,
wf.step_status,
wf.assignee,
wf.assignee_id_num,
wf.wf_start_form_field,
wf.wf_start_form_id,
wf.started_on,
wf.wf_end_form_field,
wf.wf_end_form_id,
coalesce(wf.wf_start_form_id,wf.wf_end_form_id) form_id_start_end,
wf.completed_on,

RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON asc,
wf.COMPLETED_ON asc,wf.WORKFLOW_STEP_ID asc) RANK_ASC,
RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON desc,
wf.COMPLETED_ON desc,wf.WORKFLOW_STEP_ID desc) RANK_DESC

from lbs_mv_asc_workflow_ranking wf
cross join date_para dp

where
wf.person_id <> 500000
--troubled families workflow
and wf.workflow_process_tf = 'ASSESS'
--remove cancelled
and wf.step_status_rank <> 6
--include only those with an end date
```

```
and wf.completed_on is not null
--only after 1st April
and(wf.started_on >= dp.p1 or wf.completed_on >=dp.p1)
)
--select * from WF_ASSESS;


,WF_CLOSURES_IDS as
(select wf_a.person_id, nvl(wf_o.wf_open_count,0) wf_open_count  from
(select distinct person_id from
(select person_id from WF_FEH
UNION
select person_id from WF_MISSING
UNION
select person_id from WF_CSE
UNION
select person_id from WF_CIN
UNION
select person_id from WF_ASSESS)
)wf_a
left outer join (select distinct person_id, count(distinct workflow_step_id) wf_open_count
from lbs_mv_asc_workflow_ranking where step_status_rank in (1,2,3,4) group by
person_id)wf_o on wf_a.person_id = wf_o.person_id
)
--select * from WF_CLOSURES_IDS;

,WF_CLOSURES_DATES as
(
select distinct person_id, id, start_date, end_date, greatest_date,
RANK () OVER (PARTITION by person_id ORDER BY  greatest_date desc, id desc)
WF_RANK_REV
from
(select wf.person_id,wf.workflow_step_id id, wf.started_on start_date, wf.completed_on
end_date, greatest(wf.started_on,wf.completed_on) greatest_date from WF_FEH wf
inner join (select person_id from WF_CLOSURES_IDS where wf_open_count = 0)c on
wf.person_id = c.person_id
UNION
select wf.person_id,wf.workflow_step_id id, wf.started_on start_date, wf.completed_on
end_date, greatest(wf.started_on,wf.completed_on) greatest_date from WF_MISSING wf
inner join (select person_id from WF_CLOSURES_IDS where wf_open_count = 0)c on
wf.person_id = c.person_id
UNION
select wf.person_id,wf.workflow_step_id id, wf.started_on start_date, wf.completed_on
end_date, greatest(wf.started_on,wf.completed_on) greatest_date from WF_CSE wf
inner join (select person_id from WF_CLOSURES_IDS where wf_open_count = 0)c on
wf.person_id = c.person_id
UNION
select wf.person_id,wf.workflow_step_id id, wf.started_on start_date, wf.completed_on
end_date, greatest(wf.started_on,wf.completed_on) greatest_date from WF_CIN wf
inner join (select person_id from WF_CLOSURES_IDS where wf_open_count = 0)c on
wf.person_id = c.person_id
UNION
select wf.person_id,wf.workflow_step_id id, wf.started_on start_date, wf.completed_on
end_date, greatest(wf.started_on,wf.completed_on) greatest_date from WF_ASSESS wf
inner join (select person_id from WF_CLOSURES_IDS where wf_open_count = 0)c on
wf.person_id = c.person_id)
)
--select * from WF_CLOSURES_DATES;

,WF_CLOSURES as
(select distinct
wf.person_id,
wf.workflow_id,
wf.workflow_step_id,
wf.workflow_step_type_id,
wf.workflow_step_type,
wf.started_on closure_date,
wf.started_on_system,
wf.completed_on_system completed_on,
wf.assignee_id_num,
wf.step_status,
cfa.text_answer Closure_reason,
CASE
WHEN cfa.question_lookup_user_code = '0.0' then 'RC1'
WHEN cfa.question_lookup_user_code = '0.1' then 'RC8'
WHEN cfa.question_lookup_user_code = '0.2' then 'RC2'
WHEN cfa.question_lookup_user_code = '0.3' then 'RC3'
WHEN cfa.question_lookup_user_code = '0.4' then 'RC7'
WHEN cfa.question_lookup_user_code = '0.5' then 'RC4'
WHEN cfa.question_lookup_user_code = '0.6' then 'RC6'
WHEN cfa.question_lookup_user_code = '0.7' then 'RC5'
else null end as CLOSURE_REASON_CODE,
--wf.RANK_UPDATED_REV,
--wf.RANK_ACTIVE_REV,
wf.RANK_COMPLETED
from lbs_mv_asc_workflow_ranking wf
inner join (select person_id from WF_CLOSURES_IDS where wf_open_count = 0)c on
wf.person_id = c.person_id
left outer join (select * from dm_cached_form_answers where template_id in (255,1646)
and question_user_code = '0f2a06a1-ac2d-4ee3-b80f-ab0f28e15059' and text_answer is not
null) cfa
on cfa.workflow_step_id = wf.workflow_step_id
where wf.workflow_step_type_id in (44, 3464)
and wf.step_status_rank = 5
and wf.person_id <> 500000
  order by wf.started_on
)

--select * from WF_CLOSURES;

,closure_first as
(select distinct cl.*,
--RANK () OVER (PARTITION by cl.person_id,wf.contact_step_id  ORDER BY  cl.closure_date
asc, cl.RANK_UPDATED_REV desc) WF_RANK_UP_REV,
RANK () OVER (PARTITION by cl.person_id ORDER BY  cl.closure_date asc,
cl.RANK_COMPLETED desc) WF_RANK_REV,
--RANK () OVER (PARTITION by cl.person_id,wf.contact_step_id  ORDER BY  cl.closure_date
asc, cl.RANK_ACTIVE_REV desc) WF_RANK_ACTIVE_REV,
wf.id,
wf.greatest_date
--wf.is_referral
from WF_CLOSURES cl
inner join (select * from WF_CLOSURES_DATES where wf_rank_rev = 1)wf on cl.person_id =
wf.person_id and cl.closure_date >= wf.greatest_date
)

--select * from closure_first;

--legal status closure


,LS_CLOSURE_IDS as

(select wf.person_id, nvl(ls_open_count,0) ls_open_count, wf_open_count
from WF_CLOSURES_IDS wf
left outer join
(select distinct person_id, count(distinct LEGAL_STATUS_ID) ls_open_count from
LEGAL_STATUS where legal_status_end_date is null group by person_id)ls_o on
wf.person_id = ls_o.person_id
)
--select * from LS_CLOSURE_IDS;

,LS_CLOSURES as
(select distinct
la.PERSON_ID,
 la.LEGAL_STATUS_ID,
 la.LEGAL_STATUS_CODE,
 la.LEGAL_STATUS_DESC,
la.legal_status_ranked_desc,
la.LEGAL_STATUS_START_DATE,
 la.legal_status_open_status,
la.LEGAL_STATUS_END_DATE,
 la.REASON_EPISODE_CEASED,
la.REASON_EPISODE_CEASED_DESC,
nvl(la.LEGAL_STATUS_END_DATE,to_date('01/01/1900','dd/mm/yyyy'))
LEGAL_STATUS_END_DATE_no_null,
RANK () OVER (PARTITION by la.person_id  ORDER BY  la.period_of_care_ranked_asc
desc,la.poc_rows_ranked_asc desc,la.LEGAL_STATUS_END_DATE desc,la.LEGAL_STATUS_ID
desc) P_RANK
from lbs_mv_csc_lac la
inner join (select * from LS_CLOSURE_IDS where ls_open_count = 0 and wf_open_count =
0)ls_c on la.person_id = ls_c.person_id
where la.legal_status_check  = 'Include'
```

```
and (la.legal_status_end_date is not null)
)
--select * from LS_CLOSURES;

--select * from lbs_mv_csc_lac
--where legal_status_end_date is not null

,WF_FINAL_RANK as
(select distinct person_id,

RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON asc,
wf.COMPLETED_ON asc) RANK_ASC,
RANK () OVER (PARTITION by wf.PERSON_ID ORDER BY  wf.STARTED_ON desc,
wf.COMPLETED_ON desc) RANK_DESC

from(

--ASSESS WF WITH ASSESS FACTORS
select distinct person_id, started_on, completed_on
from WF_ASSESS

UNION

--FEH wf with EH CODES
select distinct
person_id, started_on, completed_on
from WF_FEH

UNION

--MISSING WF
select distinct
wf.person_id, started_on, completed_on
from WF_MISSING wf

UNION

--CSE WF
select distinct
wf.person_id, started_on, completed_on
from WF_CSE wf

UNION

--CIN WF WITH CIN FACTORS
select distinct
person_id, started_on, completed_on
from WF_CIN

UNION
```

```
--LEGAL STATUS CODES
select distinct
wf.person_id, started_on, completed_on
from LEGAL_STATUS wf

UNION

--closure step nearest to last date for wf, where no other wf is present and only one wf step
select distinct
wf.person_id, started_on, completed_on
from closure_first wf where wf_rank_rev = 1

--UNION

--legal status closures
--select distinct
--wf.person_id, started_on, completed_on
--from LS_CLOSURES wf
--where P_RANK = 1
)wf
)


select * from WF_FINAL_RANK;

,WF_FINAL_IDS as
(select distinct person_id
from(
--ASSESS WF WITH ASSESS FACTORS
select distinct person_id
from WF_ASSESS wf

UNION

--FEH wf with EH CODES
select distinct
wf.person_id
from WF_FEH wf

UNION

--MISSING WF
select distinct
wf.person_id
from WF_MISSING wf

UNION

--CSE WF
select distinct
wf.person_id
```

```
from WF_CSE wf

UNION

--CIN WF WITH CIN FACTORS
select distinct
wf.person_id
from WF_CIN wf

UNION

--LEGAL STATUS CODES
select distinct
wf.person_id
from LEGAL_STATUS wf

UNION

--closure step nearest to last date for wf, where no other wf is present and only one wf step
select distinct
wf.person_id
from closure_first wf where wf_rank_rev = 1

UNION
--legal status closures

select distinct
wf.person_id
from LS_CLOSURES wf
where P_RANK = 1
)
)

--select * from WF_FINAL_IDS;
,ADDR as
(SELECT
distinct
addr.person_id,
Addr.flat_number,
Addr.Street_number ,
Addr.Building ,
Addr.street  ,
Addr.Town ,
addr.district ,
addr.county ,
addr.post_code ,
addr.address_start_date,
addr.address_end_date,
coalesce(address_base_uprn,addr.UNIQUE_ID) UPRN,
addr.Address_type,
addr.rank_address,
addr.address_type_rank,
addr.is_display_address,
addr.is_contact_address,
addr.address,
Addr.flat_number_num,
addr.Building_num,
Addr.Street_num,
Addr.Street_number_num,
addr.post_code_error,
addr.ref_address_id

FROM lbs_mv_asc_address_all  addr
--inner join addresses_people addrp on addrp.address_id = addr1.id
inner join WF_FINAL_IDS wf on wf.person_id = addr.person_id
cross join date_Para dp
where
--addr.address_type_code not in
('RESPLAC','TEMP','WORK','OLAPLACE','BILL','PAY','PRIMPLAC')
--and
(addr.address_end_date>= dp.p1 or addr.address_end_date is null)
)

--select * from ADDR;

--select * from lbs_mv_asc_address_all


,NHS_ID as
(select distinct nhs.person_id, nhs.macs_nhs_id from lbs_mv_nhs_macs_file_main nhs
inner join WF_FINAL_IDS wf on wf.person_id = nhs.person_id
where nhs.has_nhs_id = 'NHS ID'
and nhs.nhs_id_check = 'MISSING NHS ID FROM MOSAIC')
--select * from NHS_ID;

--select * from person_names
--order by person_id



select
distinct
su.person_id,
p.pna_UID,
p.name_class_id,

p.last_name,
UPPER(p.last_name) last_name_upper,
p.soundex_last_name last_name_soundex,
UPPER(NVL(initcap(SUBSTR(trim((regexp_substr(p.last_name,'[^ ]+',1))),0,35)),'Exclude - Surname - Blank')) AS last_name_clean,
CASE
    WHEN p.last_name LIKE '%ERROR%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%UNKNOWN%' THEN 'Exclude - Surname  - Unknown'
    WHEN p.last_name LIKE '%XX%' THEN 'Exclude - Surname - XX'
    WHEN p.last_name LIKE '%1%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%2%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%3%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%4%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%5%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%6%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%7%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%8%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%9%' THEN 'Exclude - Surname - Number'
    WHEN p.last_name LIKE '%0%' THEN 'Exclude - Surname - Number'
    WHEN length(p.last_name) < 1 THEN 'Exclude - Surname - Blank'
    WHEN p.last_name is null THEN 'Exclude - Surname - Blank'
    WHEN p.last_name LIKE '%BABY%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%UNBORN%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%INFANT%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%TWIN%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%MALE%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%FEMALE%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%BOY%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%GIRL%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%DUPLICATE%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%TEST CASE%' THEN 'Exclude - Surname - Error'
    WHEN p.last_name LIKE '%ANONYMOUS%' THEN 'Exclude - Surname - Error'
    WHEN length(p.last_name) = 1 THEN 'Exclude - Surname - Initial'
WHEN p.last_name is null THEN 'Exclude - Surname - Blank'
    WHEN length(p.last_name) = 1 THEN 'Exclude - Surname - Initial'
WHEN p.last_name LIKE '%/%' THEN 'Exclude - Surname - Character'
--WHEN p.last_name LIKE '%`%' THEN 'Exclude - Surname - Character'
--WHEN p.last_name LIKE '%-%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%.%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%#%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%@%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%$%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%*%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%!%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%£%' THEN 'Exclude - Surname - Character'
--WHEN p.last_name LIKE '%%%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%^%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%&%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%(%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%)%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%[%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%]%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%{%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%}%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%;%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%:%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%~%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%|%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%+%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%=%' THEN 'Exclude - Surname - Character'
WHEN p.last_name LIKE '%?%' THEN 'Exclude - Surname - Character'
END as last_name_error_check,

p.first_names,
UPPER(p.first_names) first_names_upper,
soundex(p.first_names) first_names_soundex,
UPPER(NVL(initcap(SUBSTR(trim((regexp_substr(p.first_names,'[^ ]+',1))),0,35)),'Exclude -
First name - Blank')) AS first_names_clean,

CASE
    WHEN p.first_names LIKE '%ERROR%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%UNKNOWN%' THEN 'Exclude - firstname - Unknown'
    WHEN p.first_names LIKE '%XX%' THEN 'Exclude - firstname - XX'
    WHEN p.first_names LIKE '%1%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%2%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%3%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%4%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%5%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%6%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%7%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%8%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%9%' THEN 'Exclude - firstname - Number'
    WHEN p.first_names LIKE '%0%' THEN 'Exclude - firstname - Number'
    WHEN length(p.first_names) < 1 THEN 'Exclude - firstname - Blank'
    WHEN p.first_names is null THEN 'Exclude - firstname - Blank'
    WHEN p.first_names LIKE '%BABY%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%UNBORN%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%INFANT%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%TWIN%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%MALE%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%FEMALE%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%BOY%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%GIRL%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%DUPLICATE%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%TEST CASE%' THEN 'Exclude - firstname - Error'
    WHEN p.first_names LIKE '%ANONYMOUS%' THEN 'Exclude - firstname - Error'
    WHEN length(p.first_names) = 1 THEN 'Exclude - firstname - Initial'
WHEN p.first_names LIKE '%/%' THEN 'Exclude - firstname - Character'
--WHEN p.first_name LIKE '%`%' THEN 'Exclude - firstname - Character'
--WHEN p.first_name LIKE '%-%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%.%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%#%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%@%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%$%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%*%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%!%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%£%' THEN 'Exclude - firstname - Character'
```

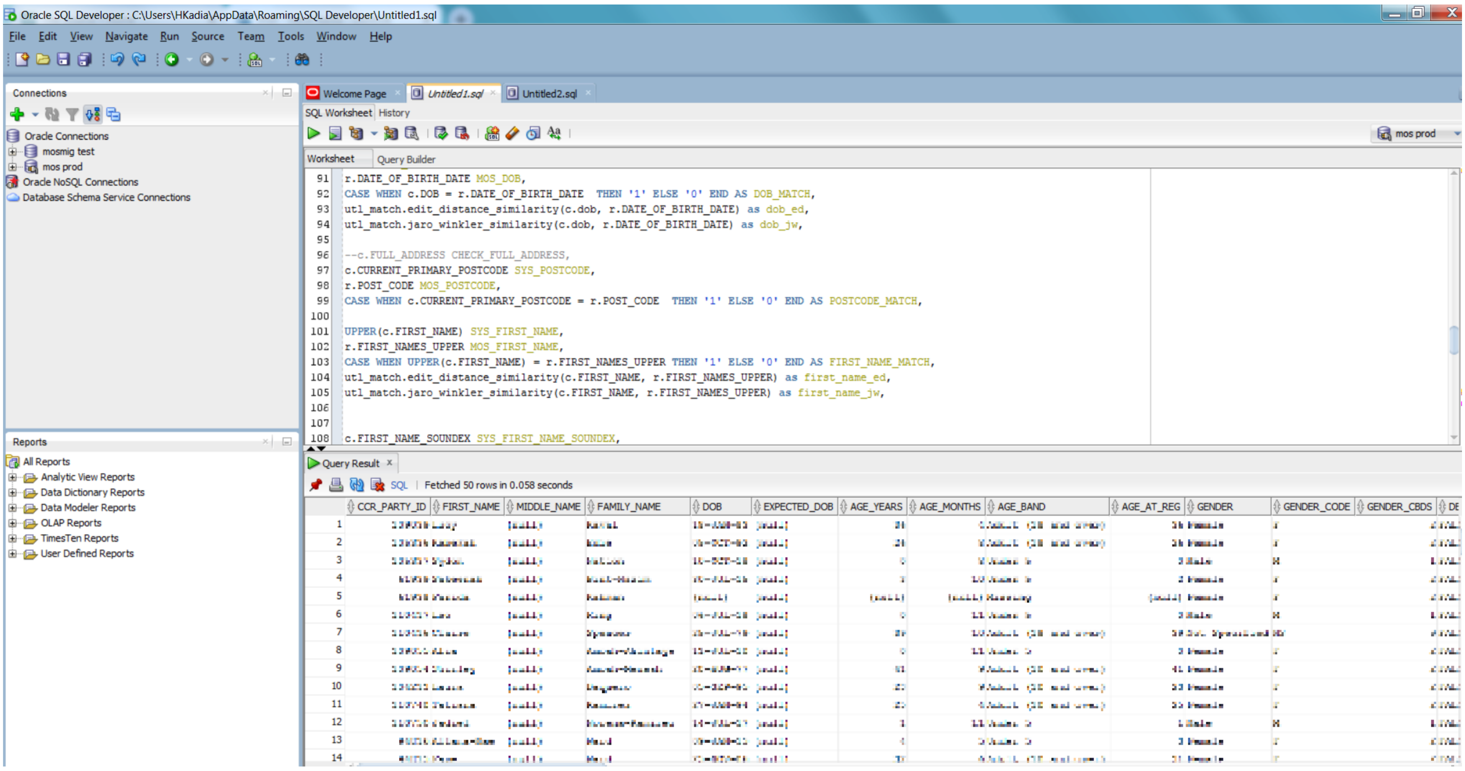
```
--WHEN p.first_name LIKE '%%%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%^%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%&%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%(%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%)%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%[%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%]%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%{%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%}%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%;%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%:%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%~%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%|%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%+%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%=%' THEN 'Exclude - firstname - Character'
WHEN p.first_names LIKE '%?%' THEN 'Exclude - firstname - Character'
END as firstname_error_check,

CASE WHEN su.title = 'Not Recorded' then null
else su.title end as TITLE,

su.DATE_OF_BIRTH_DATE,
su.DATE_OF_DEATH,

su.gender_desc GENDER,

CASE
WHEN su.NI_NUMBER = 'Not Recorded' then null
else su.NI_NUMBER end as NINO,
CASE
WHEN su.nhs_id_num is null then  nhs.macs_nhs_id
else su.nhs_id_num end as NHS_NUMBER,
CASE
WHEN su.UPN = 'Not Recorded' then null
else su.UPN end as UPN,

CASE
WHEN su.FORMER_UPN = 'Not Recorded' then null
else su.FORMER_UPN end as FORMER_UPN,

CASE
WHEN su.CAPITA = 'Not Recorded' then null
else su.CAPITA end as CAPITA,

su.cbds_ethnicity_code,
su.ethnicity_sub_ethnicity_cbds,

CASE WHEN su.religion_tf_code is null then null
else su.religion_tf_code end as RELIGION,

addr.ref_address_id,
addr.address,
addr.address_start_date,
addr.address_end_date,
addr.RANK_ADDRESS,
Addr.flat_number,
Addr.Street_number,
Addr.Building,
Addr.street,
Addr.Town,
addr.district,
addr.county,

addr.post_code,
addr.UPRN,
addr.address_type,
addr.address_type_rank,
addr.is_display_address,
addr.is_contact_address,

Addr.flat_number_num,
addr.Building_num,
Addr.Street_num,
Addr.Street_number_num,
addr.post_code_error,

--'Mosaic' "Source System",

wf_ls.legal_status_id,
wf_ls.legal_status_open_status,
wf_ls.legal_status_start_date,
wf_ls.legal_status_end_date,
wf_ls.legal_status_code,
wf_ls.legal_status_desc,

wf_feh.workflow_process_tf FEH_WORKFLOW_PROCESS,
wf_feh.workflow_step_type FEH_STEP_TYPE,
wf_feh.STARTED_ON FEH_STARTED_ON,
wf_feh.COMPLETED_ON FEH_COMPLETED_ON,

CASE WHEN wf_missing.person_id is not null then 'MISSING' else null end as
MISSING_WORKFLOW_PROCESS,
wf_missing.workflow_step_type MISSING_STEP_TYPE,
wf_missing.STARTED_ON MISSING_STARTED_ON,
wf_missing.COMPLETED_ON MISSING_COMPLETED_ON,

CASE WHEN wf_cse.person_id is not null then 'CSE' else null end as
CSE_WORKFLOW_PROCESS,
wf_cse.workflow_step_type CSE_STEP_TYPE,
wf_cse.STARTED_ON CSE_STARTED_ON,
wf_cse.COMPLETED_ON CSE_COMPLETED_ON,
CASE WHEN wf_cin.person_id is not null then 'CIN' else null end as
CIN_WORKFLOW_PROCESS,
wf_cin.workflow_step_type CIN_STEP_TYPE,
wf_cin.STARTED_ON CIN_STARTED_ON,
wf_cin.COMPLETED_ON CIN_COMPLETED_ON,

CASE WHEN wf_assess.person_id is not null then 'ASSESS' else null end as
ASSESS_WORKFLOW_PROCESS,
wf_assess.workflow_step_type ASSESS_STEP_TYPE,
wf_assess.STARTED_ON ASSESS_STARTED_ON,
wf_assess.COMPLETED_ON ASSESS_COMPLETED_ON,

CASE WHEN wf_cl.person_id is not null then 'Closure' else null end as
Closure_WORKFLOW_PROCESS,
wf_cl.workflow_step_type CLOSURE_STEP_TYPE,
wf_cl.closure_date CLOSURE_STARTED_ON,
wf_cl.COMPLETED_ON CLOSURE_COMPLETED_ON,

CASE WHEN ls_cl.person_id is not null then 'Closure' else null end as
LEGAL_STATUS_CLOSURE,
CASE WHEN ls_cl.person_id is not null then 'Looked After Legal Status' else null end as
LEGAL_STATUS_CLOSURE_TYPE,
ls_cl.legal_status_start_date LEGAL_STATUS_CLOSURE_START,
ls_cl.legal_status_end_date LEGAL_STATUS_CLOSURE_END

from WF_FINAL_IDS wf
--left outer join (select distinct wf.id, count (wf.id) count_person_id from WF_FINAL_IDS wf
group by wf.id) wf_c on wf.person_id = wf_c.person_id
left outer join lbs_mv_asc_su_demo su on wf.person_id = su.person_id
left outer join Addr on wf.person_id = Addr.person_id
left outer join NHS_ID nhs on wf.person_id = nhs.person_id
left outer join person_names p on wf.person_id = p.person_id
left outer join (select * from LEGAL_STATUS  where RANK_DESC = 1)wf_LS on wf.person_id =
wf_ls.person_id
left outer join (select * from WF_FEH where RANK_DESC = 1) wf_feh on wf.person_id =
wf_feh.person_id
left outer join (select * from WF_MISSING where RANK_DESC = 1) wf_missing on
wf.person_id=wf_missing.person_id
left outer join (select * from wf_cse where RANK_DESC = 1) wf_cse on
wf.person_id=wf_cse.person_id
left outer join (select * from wf_cin where RANK_DESC = 1) wf_cin on
wf.person_id=wf_cin.person_id
left outer join (select * from wf_assess where RANK_DESC = 1) wf_assess on
wf.person_id=wf_assess.person_id
left outer join (select * from closure_first where wf_rank_rev = 1)wf_cl on wf.person_id =
wf_cl.person_id
left outer join (select * from LS_CLOSURES where P_RANK = 1)ls_cl on wf.person_id =
ls_cl.person_id


where su.person_id <> 500000
```

```
;
--check count
select count(*) from LBS_V_CSC_TF_RISK_DQ
--44114

--create table
create table LBS_CSC_TF_RISK_DQ
as select * from LBS_V_CSC_TF_RISK_DQ

--check table
select * from LBS_CSC_TF_RISK_DQ
where last_name_error_check is not null

person_id = 285527
select count(*) from LBS_CSC_TF_RISK_DQ
--44114

--add load date
alter table LBS_CSC_TF_RISK_DQ add load_date date default sysdate;

select * from LBS_CSC_TF_RISK_DQ

--add full name
alter table LBS_CSC_TF_RISK_DQ add full_name_clean vARchar2 (150) default null;

UPDATE LBS_CSC_TF_RISK_DQ
set full_name_clean = CASE
WHEN  FIRST_NAMES is null then LAST_NAME_CLEAN
WHEN  LAST_NAME is null then FIRST_NAMES_CLEAN
ELSE  FIRST_NAMES_CLEAN||' '||LAST_NAME_CLEAN
 end;
--full name


CREATE INDEX lbs_tf_dq_fname_ix ON LBS_CSC_TF_RISK_DQ (first_names_clean);
CREATE INDEX lbs_tf_dq_lname_ix ON LBS_CSC_TF_RISK_DQ(last_name_clean);
CREATE INDEX lbs_tf_dq_funame_ix ON LBS_CSC_TF_RISK_DQ(full_name_clean);

CREATE INDEX lbs_tf_dq_fxname_ix ON LBS_CSC_TF_RISK_DQ (first_names_soundex);
CREATE INDEX lbs_tf_dq_lxname_ix ON LBS_CSC_TF_RISK_DQ(last_name_soundex);
CREATE INDEX lbs_tf_dq_dob_ix ON LBS_CSC_TF_RISK_DQ (DATE_OF_BIRTH_DATE);
CREATE INDEX lbs_tf_dq_postcode_ix ON LBS_CSC_TF_RISK_DQ (POST_CODE);
CREATE INDEX lbs_tf_dq_uprn_ix ON LBS_CSC_TF_RISK_DQ (UPRN);

CREATE INDEX lbs_tf_dq_nhs_ix ON LBS_CSC_TF_RISK_DQ (NHS_NUMBER);
CREATE INDEX lbs_tf_dq_upn_ix ON LBS_CSC_TF_RISK_DQ (upn);
CREATE INDEX lbs_tf_dq_cp_ix ON LBS_CSC_TF_RISK_DQ (CAPITA);
```

6. Match

```
WITH MATCHING_IDS as
(select
c.person_id,
r.CCR_PARTY_ID
from LBS_CSC_TF_RISK_DQ c
inner join LBS_CSC_TF_SYS_DQ r on r.DOB = c.DATE_OF_BIRTH_DATE

UNION

select
c.person_id,
r.CCR_PARTY_ID
from LBS_CSC_TF_RISK_DQ c
inner join LBS_CSC_TF_SYS_DQ r on c.POST_CODE =
r.CURRENT_PRIMARY_POSTCODE

UNION

select
c.person_id,
r.CCR_PARTY_ID
from LBS_CSC_TF_RISK_DQ c
inner join LBS_CSC_TF_SYS_DQ r on c.LAST_NAME_UPPER = UPPER(r.FAMILY_NAME)

UNION

select
c.person_id,
r.CCR_PARTY_ID
from LBS_CSC_TF_RISK_DQ c
inner join LBS_CSC_TF_SYS_DQ r on c.LAST_NAME_SOUNDEX =
r.FAMILY_NAME_SOUNDEX

UNION

select
c.person_id,
r.CCR_PARTY_ID
from LBS_CSC_TF_RISK_DQ c
inner join LBS_CSC_TF_SYS_DQ r  on c.FIRST_NAMES_SOUNDEX =
r.FIRST_NAME_SOUNDEX

UNION

select
c.person_id,
r.CCR_PARTY_ID
from LBS_CSC_TF_RISK_DQ c
inner join LBS_CSC_TF_SYS_DQ r  on c.FIRST_NAMES_UPPER =
UPPER(r.FIRST_NAME)

UNION

select
c.person_id,
r.CCR_PARTY_ID
from LBS_CSC_TF_RISK_DQ c
inner join LBS_CSC_TF_SYS_DQ r  on c.FULL_NAME = r.FULL_NAME
```

```
UNION

select
c.person_id,
r.CCR_PARTY_ID
from LBS_CSC_TF_RISK_DQ c
inner join LBS_CSC_TF_SYS_DQ r on c.UPRN = r.UPRN



)

,MATCHING_UNQ_IDS as (
select distinct person_id, CCR_PARTY_ID from MATCHING_IDS
)
--select * from MATCHING_UNQ_IDS
--where
--CCR_PARTY_ID = 93304
--and person_id =
--345020          ;

,Su_DATA as(
select
c.row_id SYS_ROW_ID,
c.CCR_PARTY_ID SYS_PERSON_ID,
r.PERSON_Id MOS_PERSON_ID,


c.FULL_NAME SYS_FULL_NAME,
r.FULL_NAME_CLEAN MOS_FULL_NAME,
CASE WHEN c.FULL_NAME = r.FULL_NAME_CLEAN  THEN '1' ELSE '0' END AS
FULL_NAME_MATCH,
utl_match.edit_distance_similarity(c.FULL_NAME, r.FULL_NAME_CLEAN) as full_name_ed,
utl_match.jaro_winkler_similarity(c.FULL_NAME, r.FULL_NAME_CLEAN) as full_name_jw,

c.DOB SYS_DOB,
r.DATE_OF_BIRTH_DATE MOS_DOB,
CASE WHEN c.DOB = r.DATE_OF_BIRTH_DATE  THEN '1' ELSE '0' END AS
DOB_MATCH,
utl_match.edit_distance_similarity(c.dob, r.DATE_OF_BIRTH_DATE) as dob_ed,
utl_match.jaro_winkler_similarity(c.dob, r.DATE_OF_BIRTH_DATE) as dob_jw,

--c.FULL_ADDRESS CHECK_FULL_ADDRESS,
c.CURRENT_PRIMARY_POSTCODE SYS_POSTCODE,
r.POST_CODE MOS_POSTCODE,
CASE WHEN c.CURRENT_PRIMARY_POSTCODE = r.POST_CODE  THEN '1' ELSE '0'
END AS POSTCODE_MATCH,

UPPER(c.FIRST_NAME) SYS_FIRST_NAME,
r.FIRST_NAMES_UPPER MOS_FIRST_NAME,
CASE WHEN UPPER(c.FIRST_NAME) = r.FIRST_NAMES_UPPER THEN '1' ELSE '0' END
AS FIRST_NAME_MATCH,
utl_match.edit_distance_similarity(c.FIRST_NAME, r.FIRST_NAMES_UPPER) as
first_name_ed,
utl_match.jaro_winkler_similarity(c.FIRST_NAME, r.FIRST_NAMES_UPPER) as
first_name_jw,


c.FIRST_NAME_SOUNDEX SYS_FIRST_NAME_SOUNDEX,
r.FIRST_NAMES_SOUNDEX MOS_FIRST_NAME_SOUNDEX,
CASE WHEN c.FIRST_NAME_SOUNDEX = r.FIRST_NAMES_SOUNDEX THEN '1' ELSE '0'
END AS FIRST_NAME_SOUNDEX_MATCH,

UPPER(c.FAMILY_NAME) SYS_LAST_NAME,
r.LAST_NAME_UPPER MOS_LAST_NAME,
CASE WHEN UPPER(c.FAMILY_NAME) = r.LAST_NAME_UPPER THEN '1' ELSE '0' END
AS LAST_NAME_MATCH,
utl_match.edit_distance_similarity(c.FAMILY_NAME, r.LAST_NAME_UPPER) as
last_name_ed,
utl_match.jaro_winkler_similarity(c.FAMILY_NAME, r.LAST_NAME_UPPER) as
last_name_jw,

c.FAMILY_NAME_SOUNDEX SYS_LAST_NAME_SOUNDEX,
r.LAST_NAME_SOUNDEX MOS_LAST_NAME_SOUNDEX,
CASE WHEN c.FAMILY_NAME_SOUNDEX = r.LAST_NAME_SOUNDEX THEN '1' ELSE '0'
END AS LAST_NAME_SOUNDEX_MATCH


from MATCHING_UNQ_IDS mi
inner join LBS_CSC_TF_SYS_DQ c  on c.CCR_PARTY_ID = mi.CCR_PARTY_ID
inner join LBS_CSC_TF_RISK_DQ r on mi.person_id = r.person_id)

--select * from LBS_CSC_TF_SYS_DQ;
--select * from LBS_CSC_TF_RISK_DQ;

select * from Su_DATA
--where
--SYS_PERSON_ID = 93304
--and MOS_PERSON_ID =
--345020
;

,Su_MaTCH_COUNT as
(select
distinct su.*,
DOB_MATCH+POSTCODE_MATCH+LAST_NAME_MATCH as
DOB_POSTCODE_LASTNAME_MATCH,
DOB_MATCH+POSTCODE_MATCH+LAST_NAME_SOUNDEX_MATCH as
DOB_POSTCODE_LASTNAME_SX_MATCH,
DOB_MATCH+LAST_NAME_SOUNDEX_MATCH as DOB_LASTNAME_SX_MATCH,
FULL_NAME_MATCH+
DOB_MATCH+
POSTCODE_MATCH+
FIRST_NAME_MATCH+
FIRST_NAME_SOUNDEX_MATCH+
LAST_NAME_MATCH+
LAST_NAME_SOUNDEX_MATCH

as MATCH_TOTAL
from su_data su)

select * from Su_MaTCH_COUNT
order by MATCH_TOTAL;
```



```
r.DATE_OF_BIRTH_DATE MOS_DOB,
CASE WHEN c.DOB = r.DATE_OF_BIRTH_DATE  THEN '1' ELSE '0' END AS DOB_MATCH,
utl_match.edit_distance_similarity(c.dob, r.DATE_OF_BIRTH_DATE) as dob_ed,
utl_match.jaro_winkler_similarity(c.dob, r.DATE_OF_BIRTH_DATE) as dob_jw,

--c.FULL_ADDRESS CHECK_FULL_ADDRESS,
c.CURRENT_PRIMARY_POSTCODE SYS_POSTCODE,
r.POST_CODE MOS_POSTCODE,
CASE WHEN c.CURRENT_PRIMARY_POSTCODE = r.POST_CODE  THEN '1' ELSE '0' END AS POSTCODE_MATCH,

UPPER(c.FIRST_NAME) SYS_FIRST_NAME,
r.FIRST_NAMES_UPPER MOS_FIRST_NAME,
CASE WHEN UPPER(c.FIRST_NAME) = r.FIRST_NAMES_UPPER THEN '1' ELSE '0' END AS FIRST_NAME_MATCH,
utl_match.edit_distance_similarity(c.FIRST_NAME, r.FIRST_NAMES_UPPER) as first_name_ed,
utl_match.jaro_winkler_similarity(c.FIRST_NAME, r.FIRST_NAMES_UPPER) as first_name_jw,


c.FIRST_NAME_SOUNDEX SYS_FIRST_NAME_SOUNDEX,
```

Fetched 50 rows in 0.058 seconds

	CCR_PARTY_ID	FIRST_NAME	MIDDLE_NAME	FAMILY_NAME	DOB	EXPECTED_DOB	AGE_YEARS	AGE_MONTHS	AGE_BAND	AGE_AT_REG	GENDER	GENDER_CODE	GENDER_CBDS	DE
1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
5	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
12	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
13	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
14	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

7. Rstudio code

library(readxl)

mosaic_data = read_excel("J:/Council/Data matching files/mosaic_data.xls")

synergy_data = read_excel("J:/Council/Data matching files/synergy_data.xlsx")

View(mosaic_data)
View(synergy_data)

options(max.print=100000)

match(mosaic_data$DATE_OF_BIRTH_DATE, synergy_data$DOB)

mosaic_data$DATE_OF_BIRTH_DATE[match(mosaic_data$DATE_OF_BIRTH_DATE, synergy_data$DOB)]

#Creating a new variable
mosaic_data$dobs=mosaic_data$DATE_OF_BIRTH_DATE[match(mosaic_data$DATE_OF_BIRTH_DATE, synergy_data$DOB)]

View(mosaic_data$dobs)

na.omit(mosaic_data$dobs)

#options(max.print=100000)

View(mosaic_data$dobs)
mosaic_data$dobs

```
mosaic_personid = mosaic_data$PERSON_ID
mosaic_data = mosaic_data[,-1]

merge(mosaic_personid, mosaic_data$dobs)

#row.names(mosaic_data) = mosaic_data$PERSON_ID
```