Group – 49: Project Report

By Shaun Paraplammoottil Cheriyan & Aswin Manoj

❖ Selection of data

This was indeed one of the more interesting and challenging projects that we have done. The first major challenge that we came across was the acquisition of a proper dataset. The reason why this was a challenge was because we needed to find a data set that fulfilled two important conditions. The first condition was that it had to be a dataset which we could understand then only we could create proper functional dependencies. The second condition was that the dataset had to have enough columns so that we could make enough functional dependencies to split it enough to create at least ten table as per the project requirement. So, we set on to find data sets of things which we had knowledge on (such as sports, space etc.) and meet the standards of the project. But unfortunately, it was an endeavour in vain as all the datasets that we found didn't have enough columns to reach the 10-table mark. There was one case in which we found a data set that fulfilled the two conditions but didn't have enough rows. So, we decided to look through random data sets in Kaggle hoping we would come across a good dataset and luckily, we came across the Zomato Dataset.

❖ Description of initial data

This data set consists of a unique id for every restaurant that Zomato delivers to as well as the restaurant name. It also gives the location of the restaurant which includes the country code, city, address, locality, and locality verbose. It also gives the various cuisines that each restaurant served. It includes monetary data such as the average cost for two, currency and price range. It provides data on whether the restaurants provide various special services (such as HasTableBooking, HasOnlineDelivery, IsDeliveringNow, Switch to Order Menu). Finally, it also includes the data on aggregate rating which is the average rating that the restaurant gets from the customers using Zomato as well as the rating text (excellent, good etc.) and rating colour (dark green, green) which were based on the aggregate rating. It was a medium sized file with a total of 9552 rows and so it met the

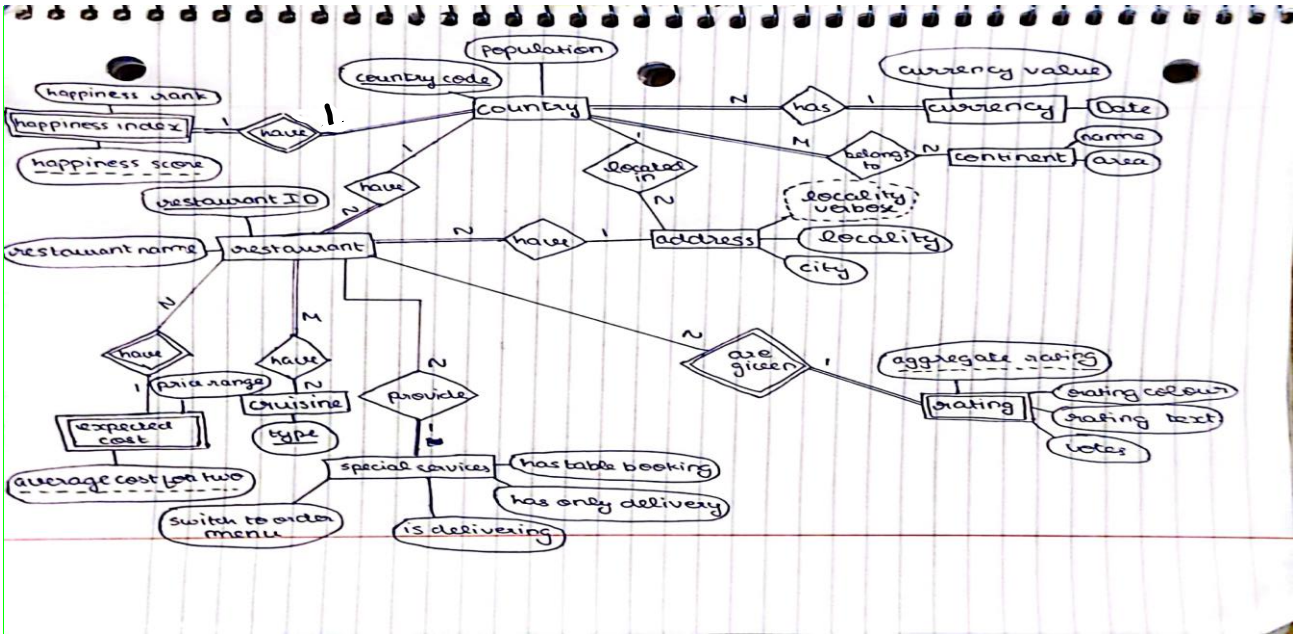
minimum requirement for the rows. More importantly, it is a dataset that is easily understandable.

- **Challenges during normalisation**

The restaurant Id was a unique value for each tuple and hence it could determine all the columns/data in the data set. Therefore, it was the primary key of the entire data set. In the first part of normalisation from 0NF to 1NF, we had a multi valued attribute cuisine as a restaurant could cook foods belonging to different cuisines. So, we separated the restaurant ID and cuisine into a separate table. After this there wasn't any multi valued attributes or compound attributes, so it was in 1NF. Fortunately, there weren't any partial dependencies and therefore there wasn't a need for any changes to be made to convert it to 2NF s it was already in that form. But for the conversion from 2NF to 3NF, there were lot of transitivity in the tables. After we removed all the transitivity, we came to understand that we unfortunately didn't have enough tables to reach the project requirements. Hence, to expand and create more tables, we merged the current data set with few other data sets. The first dataset we added was of currency values and the date (Source Kaggle) at which the currency had that value. This was made into a separate table. The second one that was added was a set of countries and the population of each country which was from the world population review website. We also added another data set from Kaggle which contained the happiness index i.e., a list of countries and their happiness scores which was calculated and compiled based on various socio-economic factors of the respective countries. It also contained the ranking of the countries in the list based on their scores. Finally, we added the list of continents and their area in square km which we were able to get from the Wikipedia website. The continents to which each country belong to was given by the world population review website. This was our final table which we added as we had reached our 10-table mark. After we finalised and got all our required datasets, we then compiled the data and properly wrote all the conversion of the data from 0NF to 3NF which was fortunately not too hard even with the addition of datasets as there was only transitivity among the new data which could be easily removed. Luckily, all the determinants were super keys and hence, no change was required to convert the data from 3NF to BCNF.

❖ Challenges during creation of ER diagram

Most of the participation constrains and cardinalities were straight forward with few exceptions. One of the exceptions was the cardinality between restaurants and address. We had initially thought that each restaurant would have its own address and it would be unique. But then, this would lead to the data set having two primary keys. However, after further inspection of the data set, we found that restaurants which would be in the same food court in a mall or in any common building all had the same address (the address of the building and the floor, which was usually the same). Since multiple restaurants could have the same address which eliminated the uniqueness of the address, we put a many to one relation between them. We All in all, the final diagram which was created was one in which we didn't have any regrets in.



❖ Creation of the Interface

Our interface involves a java code with a simple terminal that supports both the initialization of the csv tables and the commands to run specific queries. To create this interface, we had to come up with the proper syntax for the schema implementation. We had to learn some information regarding data types in SQL and how they functioned in and between table relationships. We came to know that text type is not allowed as a primary key for a table and the solution was to

just replace it with a varchar type of a specific size. As a result of doing that, we also learned that we had to maintain the same size for varchar throughout the schema for proper foreign key referencing. When it came to floating point numbers, we tried using numeric datatype with a given precision, but the code seems to fail when this was being used. Hence, we used a 'real' data type as a backup and the code worked successfully. Merging datasets and then normalizing them to create csv files for each FD was a tricky task. In the case of our cuisine dataset, it was previously a multi valued attribute in a bigger table. To make a separate table for cuisine, we had to transform, and use excel techniques on the bigger table to separate each entity in the multivalued attribute and assign it to its related restaurantID. This process had to be done 19,000 times but excel had features that automated this process. Coming back to the interface, setting it up was relatively an easy task but time consuming as it required writing insertion and creation code for 10 tables. We managed to create a button system that helped in initializing and creating tables for our database. A challenge that we faced was figuring out how to insert a csv file into a table through a coded implementation. We figured it out by referring previous assignments and using the knowledge of implementing queries through code and reusing it by replacing it with insert value statements.

As for the queries, based on our datasets, some interesting queries that we came up with are namely,

- Happiest countries for a given continent that do not use the most popular currency in that continent. This query will give us a list of the countries with the highest happiness rank in descending order in a specified continent but none of them will use the currency that is most popular in that continent.
- Top 5 restaurants with the most branches in Zomato. This query checks the total number of branches each restaurant has throughout all the countries that Zomato delivers in and returns a list in descending order based on the number of branches.
- Top 5 countries with the best average rating for restaurants in Zomato. This query will return the countries with the best average of aggregate Rating of restaurants by doing joins on restaurant, address, and country code.

Regarding the modelling of our data, I believe that modelling this data in a different way would not make much of a difference as it is already very concise and efficient.

This database could probably be modelled with a graph based or NoSQL database because the relation between each entity would not be too hard to define and there exists a lot of interconnected branches between restaurants, countries, and the other datasets. Hence, this data is not required to be run on a relational database. Complex queries on the other hand would be difficult to construct in NoSQL and graph databases especially for the ones that we defined. But the latter can be easily adjusted to changing requirements and can give faster results whereas a relational database requires a rigid schema and fore planning as it does not easily accommodate changing requirements.

- ***Sources:***

https://en.wikipedia.org/wiki/Continent

https://worldpopulationreview.com/country-rankings/list-of-countries-by-continent

https://www.kaggle.com/datasets/ruchi798/currency-exchange-rates

https://www.kaggle.com/datasets/unsdsn/world-happiness

https://www.kaggle.com/datasets/chandruuk/zomato-dataset?select=zomato.csv