**Database Documentation**

**#INTRODUCTION**

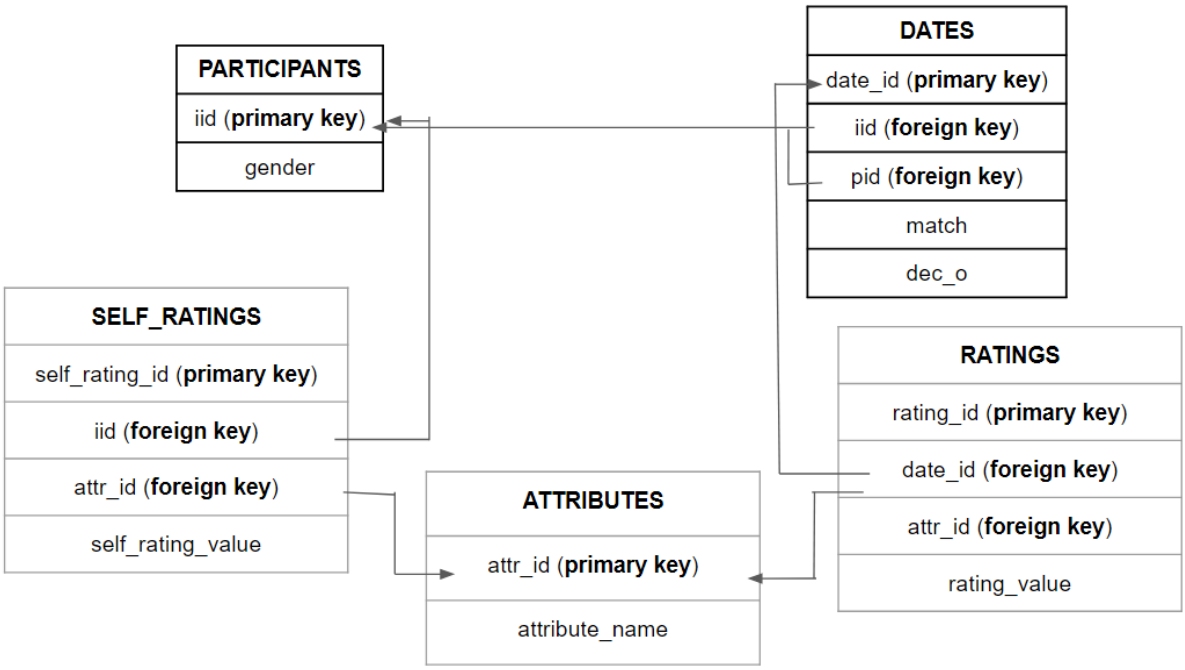
- This database imports from the Speed Dating Data.csv
- The original database concerns a speed dating experiment in which many different people participated in dates
- The study concerned five main attributes: **attractiveness, sincerity, intelligence, fun and ambition**
- Each participant would rate themselves on these five attributes (scale 1 - 10) and would also allocate 100 points across these five attributes in what they look for in a date
- After each date, each participant would rate their partner on a scale of 1 - 10 for these five attributes
- Each row in the table concerned a particular date. The columns **iid** represented a particular individual. The column **pid** represents the iid of the other individual on the date (e.g. iid's partner)
- Each row contains the gender of the person (**iid**) where 1 is male and 0 is female
- For each row, we have many values but for this study the most relevant were the columns representing the five attributes and the values that person (**pid**) rated the individual (**iid**) across these five attributes
- We had a **dec_o** column representing the decision of the partner (**pid**). If the value is 1, they agreed for a second date, if the value is 0 they did not.
- We also had a **match** column that has a value of 1 if both parties agreed for a second date and 0 if both parties did not agree for a second date

- This study aimed to find insights on the influence on the ratings received on a date for the five attributes on getting a second date (e.g. **dec_o = 1**) as well as the influence of the self ratings across these five attributes
- Therefore the columns imported into the database structure were as follows:
    - **iid -** the id representing an individual
    - **gender** - a value of 1 is male, 0 is female
    - **pid**
    - **match**
    - **dec_o**
    - Then the five attribute rating columns related to ratings received on a particular date
    - **attr_o**
    - **sinc_o**
    - **intel_o**
    - **fun_o**
    - **amb_o**
    - Then the five attribute rating columns related to self ratings
    - **attr3_1**
    - **sinc3_1**

- **intel3_1**
- **fun3_1**
- **amb3_1**

These were then split into relation tables as shown below.

**# DATABASE STRUCTURE**




- **Participants**: This contains the unique iid of each value and their respective gender (0 = female, 1 = male)
- **Dates**: This contains a date_id. The iid and pid value represent the iid values of the two people on the date. match represents whether or not they matched for this date. dec_o is the decision made by person (pid) as to whether they would go on a second date with person (iid). Therefore, for each date there are two rows. For instance, for iid = 1 and pid = 11, there will be another row where iid = 11 and pid = 1 so we can get different values for dec_o
- **Attributes**: This contains an attr_id 1-5 for each attribute. attribute_name holds the name of the particular attribute in short form for easier accessing of column names in the original database (e.g. attr instead of attractiveness)
- **Ratings**: This table is to hold the values person iid received on a particular date for the 5 different attribute values. So we get 5 rows for each attribute for each date_id. date_id links to the dates table as these ratings are per date, not per person. attr_id links to the attributes table

- **Self_ratings**: This value contains the values each person rated themselves across the five attributes. We have 5 rows for each iid representing each of the 5 attributes. iid links to the participants table. attr_id links to the attributes table

There are many reasons for this particular database structure. Firstly, this study looks at the influences of, separately, ratings given on dates and their influence on **dec_o** and ratings individuals give themselves and their influence on **dec_o** as well as the relationship between these two sets of ratings and their influence on **dec_o**.

We have ratings per individual as well as per date, hence the splitting of the **dates** and **participants** tables. The **attributes** table holds each individual attribute. This study is mainly concerned with the attractiveness attribute however all attribute values are needed in order to determine which attribute is the most influential. Additionally, similar studies could be done on the other attributes independently. The structure of this database easily accommodates more data in the form of more rows but also accommodates the adding of more attributes. The attributes are first added to the **attributes** table and the **attr_id** auto increments. The self_ratings and ratings_table are created using the attributes from the **attributes** table.

The separation of self_ratings and ratings is because we have ratings per date and ratings per individual. Splitting them this way means we have no repeated data.

Several indexes are made. These make querying the database faster but can possibly slow down writing to the database, however, we read from the database much more than we write and so this speed increase comes at a small cost of slight storage space increase. Below are the indexes:

- **idx_attributes_attr_id**
- **idx_dates_date_id**
- **idx_participants_iid**
- **idx_ratings_attr_id**
- **idx_ratings_date_id**
- **idx_self_ratings_attr_id**
- **idx_self_ratings_iid**

**# DATABASE OPERATIONS**

If you are analysing **ratings,** SQL joins can be made between **ratings** and **dates**. Allowing you to analyse ratings for a particular date as well as analysing their influence on match or dec_o outcomes. Gender analysis can easily be included within this by an additional join based on the **dates** columns iid and pid with **participants** column iid. Joins with attributes on attr_id can be used to fetch a particular attribute name (useful if you're analysing two types of attribute ratings).

The analysis of **self_ratings** is essentially the reverse of this. Again **self_ratings** can make a join with **attributes** on the attr_id column. We can analyse by gender by linking the iid columns

of both **self_ratings** and **participants**. For further information on date success, the particular iid can be linked with their respective date_ids in the **dates** table.

**# DATA PREPROCESSING AND CLEANING**

- First the relevant columns were imported from the .csv into a speed_dating table from which the relational tables were made and any empty fields were converted to NULL
- If any row had three or more NULL values for either set of five attributes, that row was deleted. If three of five values are missing it is difficult to perform data imputation based on other data.

- The database was then scanned for any missing pairs and if there were any they were deleted. For each date, we have an iid and a pid. For each date there are two rows, from perspective of iid and the perspective of pid (where pid = iid), for a clean database and to ensure relevant information, we must have both perspectives of a particular date

- The remaining NULL values were fixed using KNN (K Nearest Neighbours) data imputation with number of neighbors = 5. Averaging was considered, however the issue with this is that these are ratings of attributes. Just because you have a high ambition, you are fun and you are of high intelligence does not necessarily mean you are attractive, not everyone's *that* lucky. Currently this is just performed on columns ending in _o as these are the only columns with NULL values and improves overall efficiency. However this is easily changed for future data or a different attribute set analysis because a set of attribute columns will follow a similar structure, the first three to five letters followed by some code e.g. intel3_1 or intel3_2 and the .endswith function is used to determine which columns to perform data imputation on. If, in the future, other columns such as iid, pid, gender, dec_o or match have NULL values, these should not be included as this is vital information.

- From this complete self_ratings table, the relation tables are created. I have opted to keep the self_ratings table in there after the creation of relation tables instead of dropping it. This is because there are other sets of attributes that could be analysed in the future and we are working with a relatively small database.

**# COLUMN DATA TYPES**

Below are the data types to represent integers and the values they can hold.

- **TINYINT**: 0 to 255 (unsigned) / -128 to 127 (signed) -> 1 byte
- **SMALLINT**: 0 to 65,535 (unsigned) / -32,768 to 32,767 (signed) -> 2 bytes
- **INT** or **INTEGER**: 0 to 4,294,967,295 (unsigned) / -2,147,483,648 to 2,147,483,647 (signed) -> 4 bytes
- **BIGINT**: 0 to 18,446,744,073,709,551,615 (unsigned) / -9,223,372,036,854,775,808 - 9,223,372,036,854,775,808 (signed) -> 8 bytes

Below are the data types for each column in the relational table along with justifications.

- **iid**: SMALLINT UNSIGNED -> unlikely to have more than 65,535 participants in a dating experiment. Will never be negative but is currently more than 255
- **gender**: TINYINT -> is either 1 or 0
- **date_id**: INT -> there are two rows for each iid so worth playing on the safe side. Primary keys that auto increment must be INT
- **pid**: SMALLINT UNSIGNED -> See iid
- **match**: TINYINT -> is either 1 or 0
- **dec_o**: TINYINT -> is either 1 or 0
- **attr_id**: TINYINT UNSIGNED -> is currently 1 - 5. Can accommodate more attributes and it is unlikely that there will be more than 255 attributes. It will never be negative
- **attribute_name**: TEXT -> holds the attribute shortened name
- **rating_id**: INT UNSIGNED -> there are five ratings per iid so is INTEGER to be on the safe side. Auto Increment fields must be INT
- **rating_value**: TINYINT UNSIGNED -> currently on a scale of 1 - 10. Will accommodate new data that could be a 100 point allocation
- **self_rating_id**: INT -> ten rows per date so can mount up quickly. Primary keys that auto increment must be INT
- **self_rating_value**: TINYINT UNSIGNED -> see rating_value