Università di Pisa

Dipartimento di Informatica

Corso di Laurea Magistrale in

Data Science and Business Informatics

Relazione sul progetto di

Laboratory of data science

A cura di

Giuseppe Mirko Milazzo

Francesco Santucciu

Anno accademico 2021/2022

Assignment part 1

Pre-processing
The pre-processing phase is divided in three steps:

- The splitting of the tables and the computation of derivable attributes from the tennis.csv file (i.e. birth year of the players)
- Clean the missing values of the created tables
- Loading into the database

Get_table.py
Through the script get_table.py it is possible to perform the splitting of the tennis.csv file in the five tables tournament.csv, match.csv, date.csv, geography.csv and player.csv .
The script get_table.py works in the following way:
The core of the script is the function define_table which works in this way:

- The indices of the columns that you want to select and indices of the columns that are considered primary keys for the table that you want to create are indicated
- A reader gets initialized for the input file and then a writer is initialized for the output file: when the reader reads the header, the writer writes the names of the selected columns; on the contrary if the reader reads a record of the file, first it does a check of the indices of the columns to see if they are already inserted in the output file through a set and if the index of the value is not in this set, then it writes the record in the output file, otherwise it does not write it.

For the table tournament, the chosen columns as indexes are “tourney_id”, for the table match “tourney_id” and “match_number”, for the table date.csv “tourney_date” and for the table geography “winner_ioc” and “loser_ioc”. For the player table it was used an ad-hoc function.
Once we got these tables it was necessary make some small transformations for each of them, in particular:

- For the table match it was concatenated tourney_id and match_number.
- For the table date the format yyyymmdd was changed in such a way to split the year, the month and the day and through the function get_quarter, given in input a month of the year, it is possible to get the quarter.
- For the table geography through the function transform_geo, the file geography.csv was transformed which contains only the tuples winner/loser_ioc and through files countries.csv and country_list.csv it was performed the correspondence with the ioc in this way it was possible to derive the name of the continent and the language, after that it is possible to get the file geography.csv. For the cleaning of all the errors in the table refer to the next section
- For the table player it was used a function called define_player_table which basically works in this way:
  if both the winner or loser id is already in the player set , it is not considered , otherwise it is inserted in the output file. Through the function transform_player_table it was possible to determine the gender of the player using the assumption that through the name it is possible to determine the gender, however is there is a bisex name, then the name and the surname of the player is searched in the male player table, if it is not there, then the player is female. In order to determine the birth year, we subtracted the tournament year with the average_age. In the end,

through the function clean_missing all the missing values are substituted with a special character ‘?’.

Data_cleaning.ipynb

The file tournament.csv presents missing values only for the attribute surface, since it is only 1,27% of the data, we imputed these values with their mode.

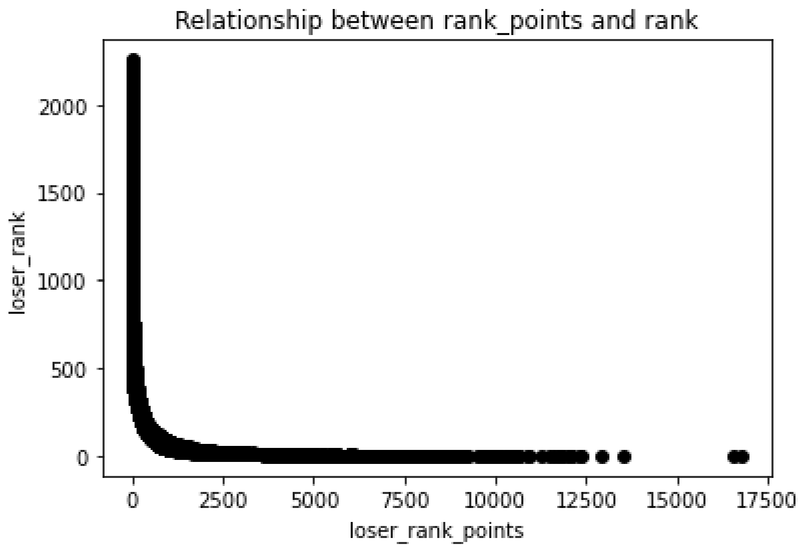
For the table match, all the attribute with a proportion of missing values higher than 50% were deleted, in this way, the only attributes with a presence of missing values are:

- score
- winner/loser rank
- winner/loser rank points

The attribute “score”, as it has a proportion of the 0.09% of missing values, these values were substituted with their mode.

The attribute winner/loser rank points, we assumed that for each tournament the values belongs to a normal distribution, so taking the mean and the standard deviation we substitute the missing values with a realization of their normal distribution.

We noticed that winner/loser rank, has a strong correlation with winner/loser rank points and this is proven by the fact that the Spearman correlation index is -0.99



We trained a random forest regression. We chose this model because it showed a R2 score of 0.97, and because the model fits well with nonlinear dependency cases.

In the end, the table player presents missing values only for the attributes:

- hand
- height
- byear

Since “height” has almost a proportion of 95% of missing values, we dropped it, on the contrary “hand” has only 0.32% missing values, so we substitute them with their mode.

For byear, it was originally decided to retrieve the missing data through web scraping[1], but since the task required dynamic scraping for each player search, we decided to see if there was any dependency that we could exploit as we did with the winner/loser rank, but non was found, so we simply dropped the rows with missing values and then made sure to remove from the match table all the winner/loser IDs where these players with missing byear appeared.
The file data_cleaning.ipynb give as output the files match_cleaned.csv, player_cleaned.csv and tournament_cleaned.csv
The phase of the analysis of the dataset ‘countries’ was performed with pandas: il dataset present any missing values, however for the countries “Netherlands”, “Nigeria”, “Germany”, “Singapore”, “Greece” and “Philippines” there ware two different code which represent the same country: it was decided to not perform any modification to these records.
The correctness of the geographical information and the syntactical correctness of the codes in the dataset was verified comparing them with a open source dataset from github[2]:
The language was extracted from the same source, which in most of the records, had more than one option since in a country, more than one language can be spoken
It was decided to modify this column using regular expressions in such a way as to obtain only the first language present for each record, as the official language.
The missing values in the language column reguarded only the contries Singapore”, “Germany”, “Greece”, “Kosovo”, “Montenegro”, “Netherland”, “Nigeria”, “Philippines”, “Trinidad and Tobago”: these values were manually substituted.
The country with the code “POC” indicates a series of small island states located in the Pacific Ocean where different languages are spoken and athletes from these islands compete under the name of the same state, Pacific Oceania: for this record (which corresponds to three players in the player table) it was decided to assign the value “Unknown” as it is not possible to define a unique language and it was decided not to assign the language spoken by the majority since for all other countries the official language was chosen.

load_data_db
This python script takes the cleaned version of the tables and load the data in the database, loading first the dimension of the datamart and at the end the fact table.

Assignment part 2

SSIS
The solution of the first assignment perfroms the connection to the database and loads the IDs of the winners and the IDs of the matches from the match table, after that it groups the players; the count of the winner_id is our measure, then it performs a lookup on the player in order to get from the winner_id the country_ioc and in the end it sorts by country_ioc.

[1] https://www.ultimatetennisstatistics.com/
[2] https://github.com/datasets/country-codes/blob/master/data/country-codes.csv

For the second assignment, it was performed the connection to the match table, after that it was performed a lookup both for the winners and the losers, in this way it was possible to get the byear
It was derived the column of the absolute value of the difference of the byear between winners and losers and applying a conditional split it was selected all those matches where ths difference was grater o equal to 6. To be able to condense in one single column the couple winner_id and loser_id, it was used a multicast and then an union all on the IDs, in this way it is possible to get a single column colled player_id. It was performed a lookup by tourney and then for date in such a way as to obtain the year and in the end we aggregated by year and by player and as a measure of aggregation it was used the count. Finally, we sorted by year and then by count in descending order.

For the third assignment, in order to transform the couple winner_id and loser_id in a single column, we used the multicast and then the union all operator. Then a lookup on the tournament table in order to get the number of spectators, we aggregated by player's IDs and as a measure we used the sum over the number of viewers. In order to get the player's names we used the lookup on the player table
And finally we sorted by number of spectators in descending order.

Assignment part 3

Data cube
It was decided to create the dimensions date, player, geography and tourney for the data cube.
In player dimension it was created the hierarchy geography which is composed by:

- continent -> country_ioc -> player_id

We created two hierarchies in tourney dimension:

- DayMonthQuarterYear composed by year -> quarter -> month -> date_Id
- YearTournament composed by tourney_name -> tourney_id, in particular with this last hierarchy it is possible to drill-down on the tournment and its different annual editions.

Moreover, in order to create the hierarchy DayMonthQuarterYear in a proper way, it was computed two attrobutes which are "the quarter" and "the month" obtained with the following SQL commands:

```
case month
when 1 then 'January'
when 2 then 'February'
when 3 then 'March'
when 4 then 'April'
when 5 then 'May'
when 6 then 'June'
when 7 then 'July'
when 8 then 'August'
when 9 then 'September'
when 10 then 'October'
when 11 then 'November'
else 'December' end
```

```
case quarter
when 1 then 'Q1'
when 2 then 'Q2'
when 3 then 'Q3'
else 'Q4' end
```

In this way we could give an order to the months and to the quarters

Queries
To compute the increament of the loser rank points with resperct to the previous year, it was used the following formula:

((rank_points_CurrentYear * 100) / rank_points_PreviousYear) – 100)

In the second query, to calculate the percentage of winner_rank_points for the annual edition of the tournament compared to the total winner_rank_points of all editions of the tournament, we leveraged the YearTournament hierarchy to calculatethe overall total of winner_rank_points for all editions of the tournament. Once we obtained this total, we used it as the denominator in the ratio:

(rank_points_torneoCorrente / tot_rank_points_edizioniTorneo)

For the third and final query, instead of using the AVG() function, it was decided to calculate it manually by obtaining the sum of winner_rank_points for the player's continent for all years and the dividing it by the count of matches played by players from that continent for all years

<u>Chosen dashboard:</u>

It was decided to plot the distribution of gender participation in tennis matches in each continent/country for each year. This was considered interesting as it allows for observing whether there are gender differences in tennis in different countries around the world from 2016 to 2021. The chosen measure was the count of matches.