University of Pisa

Master's Degree in

Data Science and Business Informatics

Department of Computer Science

Laboratory of Data Science

Gianni Andreozzi

Martina Trigilia

Academic year 2021/2022

Index

1 Part One

1.1 Pre Processing

The *Tennis* dataset consists of 186,073 records and 49 variables and contains data on various tennis matches, with some information regarding the tournament, date, winner and loser, respective scores, and more.

One problem we immediately had to deal with was that of player, tournament, and match ids. As far as the players were concerned, there were some entries in the tennis dataset which, although they had the same player name, had a different player_id associated with them; although we could think of possible homonyms, by checking on some of them we were able to verify that these players had different ids associated with them in the various rows. It was therefore decided to assign a single id to these players. In other cases, on the other hand, the same player_id was referred to by more than one player name: we therefore proceeded to assign new ids to these players, generating them from the maximum player id in the table. We also encountered similar problems with the tournament ids: there were some entries in the dataset that were associated with the same tourney_id, but had different tournament names. An example is the tournament '2017-M009' which is once associated with the name 'Rome Masters' and once with the name 'Rome'. In this case, one might think of an error in reporting the name of the tournament, however, these entries not only have a different name but also a different tourney_level. For this reason it was decided to treat them as different tournaments for all intents and purposes and a new id was generated and assigned, concatenating the previous one to the string '1'. Similarly, we noticed that there were some entries with the same tourney_id and match_num (which together form the primary key of the match table), but played by different players (so they were not duplicates). These were also treated as different matches and a new id was generated for them by concatenating the string '20' to the duplicate match number. No further inconsistencies were found.

Afterwards, some data cleaning operations were carried out. Several errors were present in the attribute concerning the IOC code. These had to be corrected as information such as the nationality of the players may be important for future analyses. For example, some players had the code 'DEU', while others had the code 'GER': in cases like the latter, the same code was entered for all players (following the list of IOCs for the various nationalities found on the official Olympic website). Players with code

IOC = 'POC' are actually players for whom the nationality is missing and the correct code was entered for these. Other errors, mainly human and spelling errors, were found and corrected in the countries.csv table and the country_list.csv table used to find the country language. In addition, there were 38 IOC codes in the tennis.csv table that did not appear in the countries.csv table. We did not find any other tables on the Internet that were considered reliable and complete to be able to do data integration of these missing codes, so for the following, the relevant name and continent were found (https://en.wikipedia.org/wiki/List_of_IOC_country_codes) and inserted into the countries.csv table. Further changes were also made to the female_players.csv and male_players.csv files, as the names within them contained spelling errors and it was therefore not possible to assign a gender to all players in tennis.csv. Please note that all pre-processing changes made to the files were made using the pandas library.

1.2 Missing Values

Each variable was analysed individually to determine the best approach for the possible imputation of missing data. Table 1 shows the number of missing data for each column. For each variable, the missing information was initially searched in other records for matching ids, and if found, was subsequently imputed. *Surface* is a categorical variable, to replace the missing data in it, the probabilities of observing each of the 4 surfaces expressed as cumulative frequencies were calculated, a value was then generated from a uniform distribution [0,1] for each tournament (it is assumed that the surface does not change within a tournament). The surface was chosen by observing the range defined by the cumulative frequencies in which the value falls, in order to keep the initial proportion as unchanged as possible. Figure 1 shows the distribution of the surfaces. With regard to *winner_hand* and *loser_hand*, it is assumed that a player can be either right- or left-handed. The value 'U' was considered as missing. Studies report that about 15% of professionals are left-handed (Francois Fagan et al. 2018), so the substitution was performed by generating for each player a number from a Bernoulli distribution with probability of success equal to 0.15 and putting him left-handed in case of success. The missing attributes of the *score* variable were eliminated, as they affect few records. For height, the missing data were replaced with the median, stratifying by gender and country of origin. This resulted in a shift to the left of the overall distribution as shown in figure 2. This is mostly due to the fact that missing values were more frequent among females, in fact, only 2.59% of the female players had height, while the percentage is

12.64% for males. The distribution of minutes had a strong positive skewness, so the median was used to replace the missing values. The same applies to the variables *w_ace* to *l_bpFaced*. As for the players' rank, according to the official ATP World Tour website, it is updated every 52 weeks (approximately one year) taking into account the score totalled during that period. For this, we searched for the variables *winner_rank and loser_rank* by corresponding player_id and for the same year. In contrast, for the variables *winner_rank_points* and *loser_rank_points it* was not considered appropriate to replace the missing values in any way: the score of a player in a game is influenced by too many factors to be imputed, such as the tournament in which he plays, the phase of the tournament and the qualification mode for the individual player (https://www.atptour.com/en/corporate/rulebook). The missing values of the attribute winner_entry and loser_entry were not dealt with, as this is superfluous for the creation of the necessary tables. The missing values of the columns winner_ht, winner_age, loser_ht, loser_age, winner_rank, winner_rank_points, loser_rank, loser_rank_points were replaced with the value -1, in order to facilitate their subsequent loading into the database. The resulting file has 185,595 rows.

Variable	***Missing Values Pre***	***Missing Values Post***
W_bpSaved	103789	0
W_bpFaced	103789	0
L_ace	103789	0
L_df	103789	0
L_svpt	103789	0
L_1stIn	103789	0
L_1stWon	103789	0
L_2ndWon	103789	0
L_SvGms	103785	0
L_bpSaved	103789	0
L_bpFaced	103789	0
Winner_rank	19379	19344
Winner_rank_points	19395	19360
Loser_rank	35256	35206
Loser_Rank_points	35272	35222

Variable	***Missing Values Pre***	***Missing Values Post***
Tourney_spectators	0	0
Tourney_revenue	0	0
Tourney_id	0	0
Tourney_name	0	0
Surface	162	0
Draw_size	0	0
Tourney_level	0	0
Tourney_date	0	0
Match_num	0	0
Winner_id	0	0
Winner_entry	160251	159794
Winner_name	0	0
Winner_hand	16	0
Winner_ht	136745	13682

Variable	*Missing Values Pre*	*Missing Values Post*
Winner_ioc	0	0
Winner_age	2828	*2810*
Loser_id	0	0
Loser_entry	141927	141927
Loser_name	0	0
Loser_hand	71	0
Loser_ht	147729	15277
Loser_ioc	0	0
Loser_age	6502	6481
Score	175	0

Variable	*Missing Values Pre*	*Missing Values Post*
Best_of	0	0
Round	0	0
Minutes	104437	0
W_ace	103789	0
W_df	103789	0
W_svpt	103789	0
W_1stIn	103789	0
W_1stWon	103789	0
W_2ndWon	103789	0
W_SvGms	103785	0

Table 1 - Missing values per variable before and after pre-processing

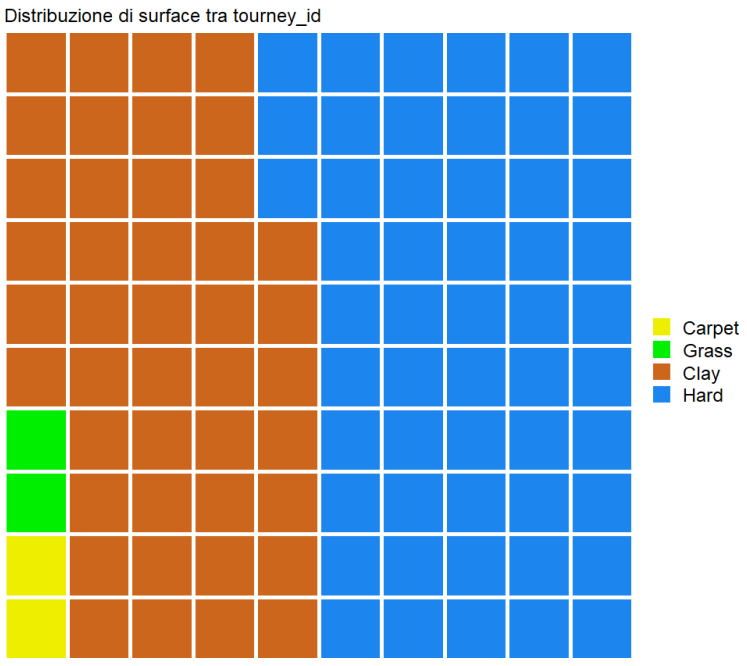

Figure 1 - Distribution of post-replacement missing values

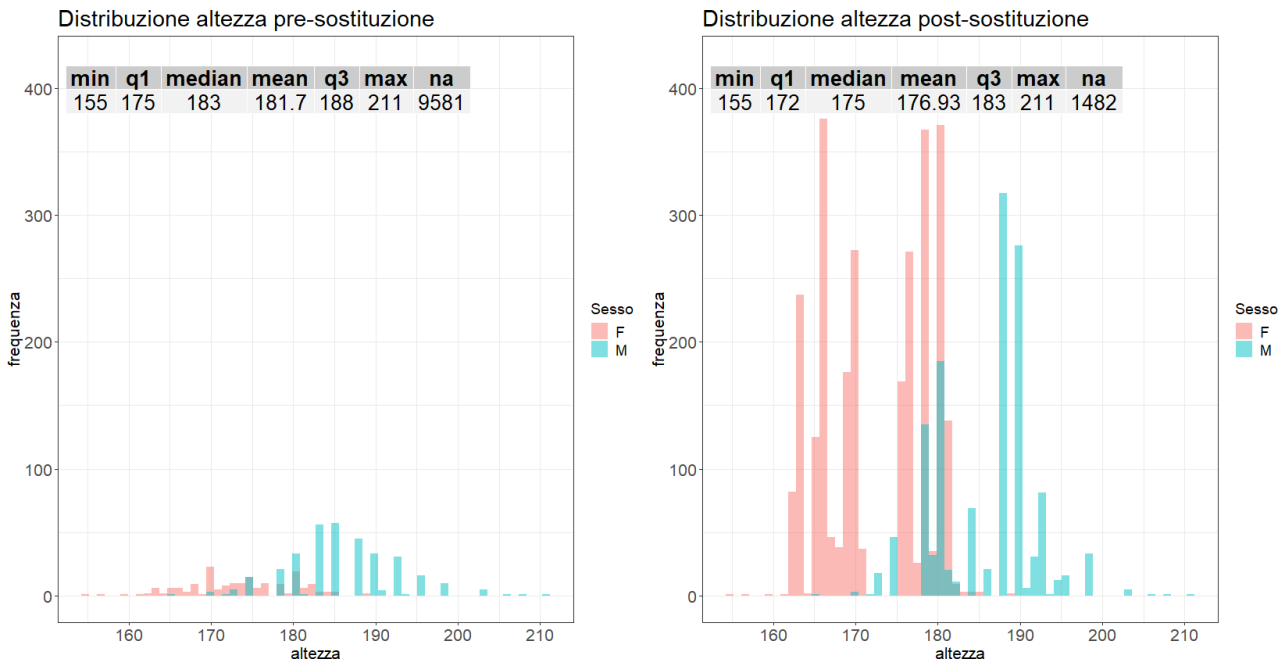

Figure 3 - Height distribution per player divided by gender pre/post replacement missing values

1.3 Code Structure

The code is structured in two files: Assignment_1 and Assignment_2.

- The **Assignment_1** file splits the tennis table into the Match, Tournament, Date and Player tables. All csv files are read with the DictReader function. The function 'create_tables' initially creates the structures of all tables, i.e. it writes their headers and initialises sets (tourney_date_ids, players_ids and tourney_date_ids), which are used to keep track of the ids used, guaranteeing their uniqueness, so that the same id is not entered more than once within the respective table. In addition, it creates the Geography table: information about the respective country's ioc code and continent is taken from the country table, while another table, country_list.csv, is used to obtain information about the language used. Subsequently, through a loop, it reads each row of the tennis table and for each of these rows, functions are called that create the corresponding rows of the Match, Player, Date tables. As far as the function that creates the row for Match is concerned, no particular manipulations were made since all the entries in the tennis table represent a match relating to a tournament, and so the match_id column was simply created from the concatenation of the tennis match_num and tourney_id columns. In the function that creates the row for the Date table, t, the date is transformed to datetime and the

information concerning the year, the month, the day and, via the function, get_quarter, the quarter. Instead, , the function to create the player row performs two operations to extract information concerning the gender and year of birth of the corresponding player. Initially, by means of the map_player function, a dictionary is created which associates with the string "male" a list containing all the names within the male_players file, and with the string "female" associates, instead, those in the female_players file; the choice of using a dictionary was dictated by the need to increase the efficiency of the sex retrive operation. To obtain the player's year of birth, it is not sufficient to subtract his age from the year of the tournament, since the player's age is not an integer, but a float whose decimal part represents the fraction of the year elapsed since his last birthday, let us call it y. We then calculate the distance between the date of the tournament and the end of the year, let's call it x, and compare this value with 1-y, i.e. the days until the next birthday. If $x \leq y$ then the player has a birthday before the tournament date. In this case

you simply do *tournament year - player years*, otherwise you do *tournament year - player years -1*.

- The **Assignment_2** file performs the loading of the Match, Tournament, Date, Player and Geography tables within the "Group_17_DB" database, respecting the relationships between the tables as established within the relational schema. Note that five commits are performed during the loading of the fact table to facilitate the process.

2 Part Two

For the second part of the project, a single SSIS project was created containing three packages, each containing the resolution of the corresponding assignment.

2.1 Assignment 0

"For every tournament, the players ordered by number of matches won."

To answer this question, it is necessary to read the Match table, in particular the columns tourney_id and winner_id, since players who have not won any matches within a given tournament should not appear in the report. By means of an Aggregation node, it was possible to perform a group by on tourney_id and winner_id and subsequently, with an Sorting node, a ranking of the players by matches won for each tournament was compiled.

2.2 Assignment 1

"A tournament is said to be 'worldwide' if no more than 30% of the participants come from the same continent. List all the worldwide tournaments."

To proceed to solve this assignment, we need to know, for each tournament, the number of distinct players who participated in it ("Distinct_Player_Per_Tourney"), the continent from which the most players came, and its frequency ("MaxDistinctContinent"). Thus, if for any tournament it is true that MaxDistinctContinent * 100/ Distinct_Player_Per_Tourney is less than or equal to the desired threshold, then we will say that the tournament in question is "WorldWide" and will therefore be displayed in the final result.

Distinct Players Per Tourney → The Match table was accessed by reading the columns tourney_id, winner_id and loser_id. Via the UnPivot node, the total set of winners and losers was obtained. Via a union node, duplicate players were removed from the set. The result is distributed over two different outputs via a MultiCast node. Subsequently via an Aggregation node, the number of distinct players per tournament is obtained.

Max Distinct Continent → Via two Search nodes, two joins were made in order to add the columns country_id (from the Player table) and continent (from the Geography table) respectively. Via a Join node, the number of players who participated in each tournament and on each continent is obtained (distinct_cont). Another Aggregation node is used to obtain the maximum for the distinct_cont column for each tournament.

A Join is then made on the two data streams obtained in the two previous steps. Via a Derived Column node, a new column (Max_Percent) is created containing the calculation Max_distinct_cont * 100 / distinct_playerPerTourney. Finally, via a Conditional Subdivision node, tournaments are taken for which it is valid that Max_Percent <= 31.

After the pre-processing carried out in the first part of the project, there do not appear to be any tournaments in the dataset for which the number of players from the same continent is less than or equal to 30%. For this reason, we decided to raise the threshold for which a tournament is called 'WorldWide' to 31%.

2.3 Assignment 2

"For each country, list all players that won more matches than the average number of won matches for all players of the same country."

To answer the previous question, we calculated the number of wins for each player and the average number of wins won for players belonging to the same continent. The Match table (winner_id) is read and, using an Aggregation node, the number of matches won for each winner_id (matches_won) is obtained. A Search node is used to add the columns country_id and name (from the Player table). A Multicast node is used for the next step which involves calculating the average number of wins per country_id (avg_matches_won_per_country), via a Merge Join node, and comparing it with the matches_won of each player belonging to that country_id. By means of a Merge Join we make a join on the two data streams and thus have a table containing one row per player and the following columns: player_id, name, country_id, matches_won, avg_matches_won_per_country. As a final step, a conditional split node is used to find the players that meet the requirement.

3 Part Three

In the last part of this project, we created a datacube via SSAS and, using the appropriate queries, answered the questions that were proposed. Finally, two dashboards created using Power BI and contained on two different pages (Assignment 4 and Assignment 5) of the same report are proposed for solving the last two assignments.

3.1 Assignment 0

After establishing a connection to the database via http protocol, the previously created DW was set as the data source. Next, the cube was created with the rank and rank points measurements of losers and winners. Two dimensions were created, Player and Tournament. Within the Player dimension, we inserted the CountryContinent hierarchy (Continent → Country_ioc), with the attributes of the Geography table. Furthermore, the derived column The Month was added to the Date table, representing the month of the year, and the functional dependency The Month → Month was defined. The hierarchy MonthQuarterYear (Year → Quarter → The Month → Date Id) was then inserted, in the Tournament dimension, using the attributes of the Date table.

3.2 Assignment 1

"Show the percentage increase in winner rank points with respect to the previous year for each winner".

To answer this question, winner_rank_points and *perc_incr* were selected on the columns, and on the rows the tuple containing the player's name and the current year. The *perc_incr member* was calculated as *diff/abs_incr.* The member *diff* is given by the difference between the winner_rank_points of the tuple (player, year) considered and *abs incr; the* latter member represents the winner_rank_points of the same player but totalled in the year preceding the one considered, thanks to the function *currentmember.lag(1)*. For all those players whose *abs_incr* is zero, a "-" is displayed, since we cannot speak of an increment in that case. Furthermore, we can see that many players in different years show a negative increment.

3.4 Assignment 2

"For each country, show the total winner rank points in percentage with respect to the total winner rank points of the corresponding continent"

We have selected winner_rank_points and *perc_country on the columns,* and on the rows the tuple containing the country_ioc and its corresponding continent. The *perc_country* member is a fraction in percent where on the numerator we have the winner_rank_points of the country in question and on the denominator that of its continent, calculated currentmember.parent.

3.5 Assignment 3

"Show the losers having a total loser rank points greater than 10% of the totals loser rank points in each continent by continent and year"

We selected loser_rank_points, *ratio* and *points_contyear on the* columns, and on the rows the tuple containing the year, continent and player name, filtered by ratio > 0.10. The *ratio* member is a fraction in percentages where at the numerator we have the loser_rank_points of the tuple considered and at the denominator *points_contyear*, i.e. the total loser_rank_points per continent and year, calculated via current member.parent.

3.6 Assignment 4

Three graphs are presented in the dashboard to show the geographical distribution of winner_rank_points and loser_rank_points. The two bar graphs show the distribution of the two measures per continent and offer the possibility to drill down to observe the distribution per country within the individual continent

overall. The third graph is a scatterplot that allows the two measures to be observed simultaneously and offers the same possibilities as the bar graphs. The latter graph has the disadvantage that it shows differences between countries and continents less clearly.

3.7 Assignment 5

Three graphs are presented in the dashboard. The first shows the trend over time of winner_rank_points divided by continent and offers the possibility to drill down and see in detail quarter and month for each year. The other two show the top ten players by winner_rank_points. One shows the continent of origin of the players while the other shows how the points earned by each player are divided by surface type.