UNIVERSITÀ DI PISA

LABORATORY OF DATA SCIENCE

2021/2022

Akhil Varma Dantuluri, Piyush Tada

Introduction

This paper intends to create and populate a database starting from .csv files then create SISS packages to answer questions on a database. After that, we created a cube in Microsoft Analysis Services so that we can do MDX queries on them. In the final stage of the project, we used the cube to create an interactive dashboard to show statistics we got from our cube.

Part 1

Assignment 0

Create the database schema in Figure 1 using SQL Server Management Studio in server lds.di.unipi.it

Data Preparation

We were provided a tennis.csv file and the assignment was to split the content of tennis.csv into four separate tables: match, tournament, date, and player. And to create the attribute" sex" for the player table Using the files male players.csv and female players.csv.
And the process is to first create a DB. Then create a table. Then populate it with data.

Assignment 1

Write a python program that splits the content of tennis.csv into four separate tables: match, tournament, date, and player. Use the files male players.csv and female plauyers.csv to create the attribute "sex" for the player table. The use of the pandas' library is forbidden for this assignment.

We have first checked the columns of the files if we find any missing and null values, at first, we have processed the file tennis i.e., checked the column names. And then we have created an output file to store the data. The header was written to the output file and following it we had to insert the date of birth for both the winner and the loser and following it we had to add a year of the birth row to the output file and, we have added an information row to the output file.

Creating Player's Table

We have created a plyers table with headers 'player_id', 'country_id', 'name', 'sex', 'hand', 'ht', 'byear_of_birth'. which has all the entries of both winners and losers we had before.

We have used the below equation to calculate the birth year of both winners and losers.

$$year\ of\ birthwinner\ =\ tourney\_date[:4]\ -\ winner\_age$$
$$year\ of\ birthwinner\ =\ tourney\_date:4\ -\ winner\_age$$

$$year\ of\ birth\ loser\ =\ tourney\_date[:4]\ -\ loser\_age$$
$$year\ of\ birthloser\ =\ tourney\_date:4\ -\ loser\_age$$

And then we have created a plyers list combining both winners and losers lists and putting sex values according to sheet names, we have included the headers like Name, Surname, Sex, and Full name.
For the player's sex column, we have first added male players representing "m" to the table and the female players 'f'.
Following this, we inserted value in the player's dictionary to create final players' table

Create Table In Database

As a first step, we created the table required by the schema provided in the project in the database. Secondly, insert the data in CSV into the database using python script insert_data_in_table.py.

Assignment 2

Write a Python program that populates the database GroupIDHWMart with the various tables from the .csv files, establishing schema relations as necessary.

Creating Tournament Table

The tournament table was created with headers
header = ['tourney_id', 'date_id', 'tourney_name', 'surface', 'draw_size', 'tourney_level', 'tourney_spectators', 'tourney_revenue']

Creating Match Table

And following the above we have created the match table with all the formation regarding the matches, with
headers= ['tourney_id', 'match_id', 'winner_id', 'loser_id', 'score', 'best_of', 'round', 'minutes', 'w_ace', 'w_df', 'w_svpt', 'l_1stIn', 'w_1stWon', 'l_2ndWon', 'w_SvGms' ,'w_bpSaved', 'w_bpFaced', 'l_ace', 'l_df', 'l_svpt', 'l_1stIn', 'l_1stWon', 'l_2ndWon', 'l_SvGms', 'l_bpSaved', 'l_bpFaced', 'winner_rank', 'winner_rank_points' ,'loser_rank', 'loser_rank_points']

Creating Geography Table

We have created a geography table with headers like

header = ['country_ioc', 'continent', 'language'].

In the geography table, we had created the information of languages for every country which was not available in our data from the beginning to solve this issue we have used a language data source from https://www.infoplease.com/world/countries/languages-spoken-in-each-country-of-the-world which has a table of world countries with their recognized languages on the percentage of the population spoke that particular language, so many countries had more a couple of languages that were spoken to overcome this problem we have chosen the language that was spoken by most of the population or most popular one among all to every respective country.

Part 2

The output we got from the following queries was exported as a CSV file at the end of the process.

Assignment 0

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

We have created the data flow task and inserted an OLE DB SOURCE as our first node to access the match fact table as for the calculation part we have aggregated tourney_id and winner_id using the node aggregation and then we used the node sort followed by aggregation on wins in descending order and then we used flat file destination to output the source as CSV file.
The top 5 rows of the result

tourney_id	winner_id	wins
2016-7316	104945	8
2017-0363	106290	8
2017-0405	201610	8
2017-0460	105936	8
2017-0692	105641	8

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.

We have created the data flow task and inserted an OLE DB SOURCE as our first node to access the match fact table followed by a multicast node to create a single column of player_id combining both winner_id and looser_id using the union all node.

Then we used a lookup node to find the player's country_id using the player table and we used a lookup node again for the geography table to get the respective continent of the player's country.

We used aggregation node to count player_id with respective group by tourney_id and continent and we used another aggregation node for tourney_id and sum of players as total player for continent using multicast node and then we joined both the aggregation tables using merge join node joining them on tourney_id and then we used the data conversion node deriving the new columns of total players by continent and total players by tournament as float then we used derived column node to calculate the percentage of players in each tournament by continent naming it AVG.

We used aggregation node grouping by tourney_id and for a maximum of AVG and then we used conditional split node for the condition if no more than 30% of the participants come from the same continent then we used flat file destination to output the source as a CSV file.

The table below is the result we obtained.

tourney_id	AVG
2018-W-FC-2018-G2-AO-A-M-NZL-LBN-01	0.25
2019-W-ITF-RSA-01A-2019	0.274194

Assignment 2

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

We have created the data flow task and inserted two OLE DB SOURCE nodes as our first node to access match fact table and the second to access player fact table and then following we used lookup node from OLE DB SOURCE match fact table to connect winner_id as player_id from look up node we used two aggregation nodes using multicast node where in aggregate node we grouped by winner_id and country_id and counted match_id as player wins.

In aggregate 1 we grouped by country_id and count match_id as match_wins_by_country and following the OLE DB SOURCE player fact table using aggregation node we grouped by country_id and count player_id as number_of players and the we used merge join node to join aggregate 1 and 2 on country_id and then we used derived column node to calculate

average wins by country using the columns match wins by country and number of players by country and then using merge join node we joined both derived column and aggregate column on country_id and then we used conditional split node to put the condition players wins are greater than average wins by country and then sort by avg wins then we used flat file destination to output the source as CSV file.

player_id
210303
211227
105430
210786
210958

Part 3

Assignment 0

Build a datacube from the data of the tables in your database, defining the appropriate hierarchies for time and geography. Use the rank and rank points of the winner and loser as measures.

To create the cube we first made a hierarchy on geography in the player and time hierarchy in the tournament. Next, we created the cube with a wizard.

Assignment 1

Show the percentage increase in winner rank points concerning the previous year for each winner

```
with member rankpoint as
iif([Measures].[Winner Rank Points] = null, 0, [Measures].[Winner Rank Points])

member pevpoint as
([Tournament].[YearDateTourney].currentmember.prevmember, [Measures].[Winner Rank Points])

member per as
iif(pevpoint=0,0,(rankpoint - pevpoint) / pevpoint),
format_string = "percent"

select  (per, [Tournament].[YearDateTourney].[Year] ) on columns,
(nonempty([Winner].[Name].[Name])) on rows
from [Group 24 DB]
```

In this case, we first took the winner rank point then we got the previous year, then we calculated the average change, which we showed in the final results.

Assignment 2

For each country show the total winner rank points in percentage for the total winner rank points of the corresponding continent.

```
with member rankpoint as
iif([Measures].[Winner Rank Points]=0, 0, [Measures].[Winner Rank Points])

member rankpoint_continet as
([Winner].[Geography].currentmember.parent, [Measures].[Winner Rank])

member per as
rankpoint/rankpoint_continet,
format_string = "percent"

select per on columns,
[Winner].[Geography].[Country Ioc] on rows
from [Group 24 DB]
```

Same as before we took rank points and made sure if there is any nan value we replace it with 0, then we create the total by continent by moving up in the hierarchy. Then we calculated the percentage then we displayed the results.

Assignment 3

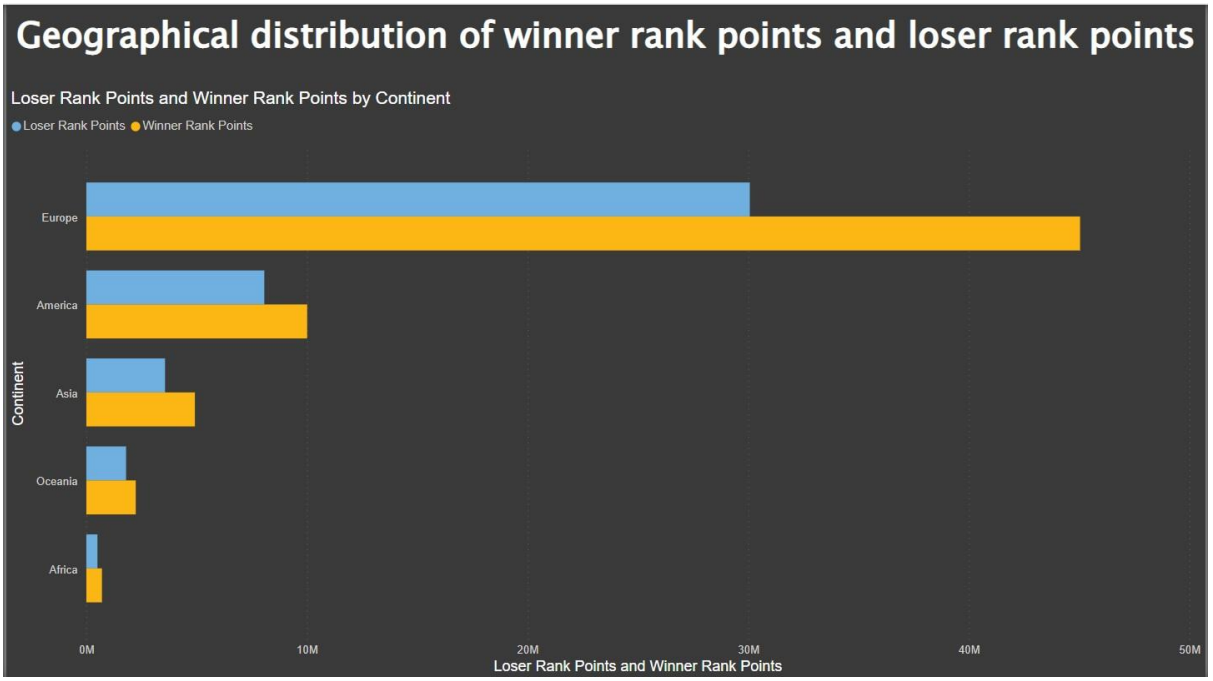
Show the losers having a total loser rank points greater than 10% of the total loser rank points in each continent by continent and year.

```
with member rankpoint_continent_year as
(
[Loser].[Name].[All],
[Loser].[Geography].currentmember,
[Tournament].[YearDateTourney].currentmember,
[Measures].[Loser Rank Points]
) * 0.1 -- getting 10% of the continent


select [Measures].[Loser Rank Points] on columns,
nonempty(
    filter
        (
            ([Loser].[Name].[Name], [Loser].[Geography].[Continent], [Tournament].[YearDateTourney].[Year]),
            [Measures].[Loser Rank Points] > rankpoint_continent_year

        )) on rows
from [Group 24 DB]
```

In the above case, we wanted to first calculate the loser rank point by continent and year, so we calculated it with a member. Then to remove empty entries we used a nonempty function, then we used a filter to get only players with loser rank points more than 10% of total looser rank points by continent and year.

Assignment 4

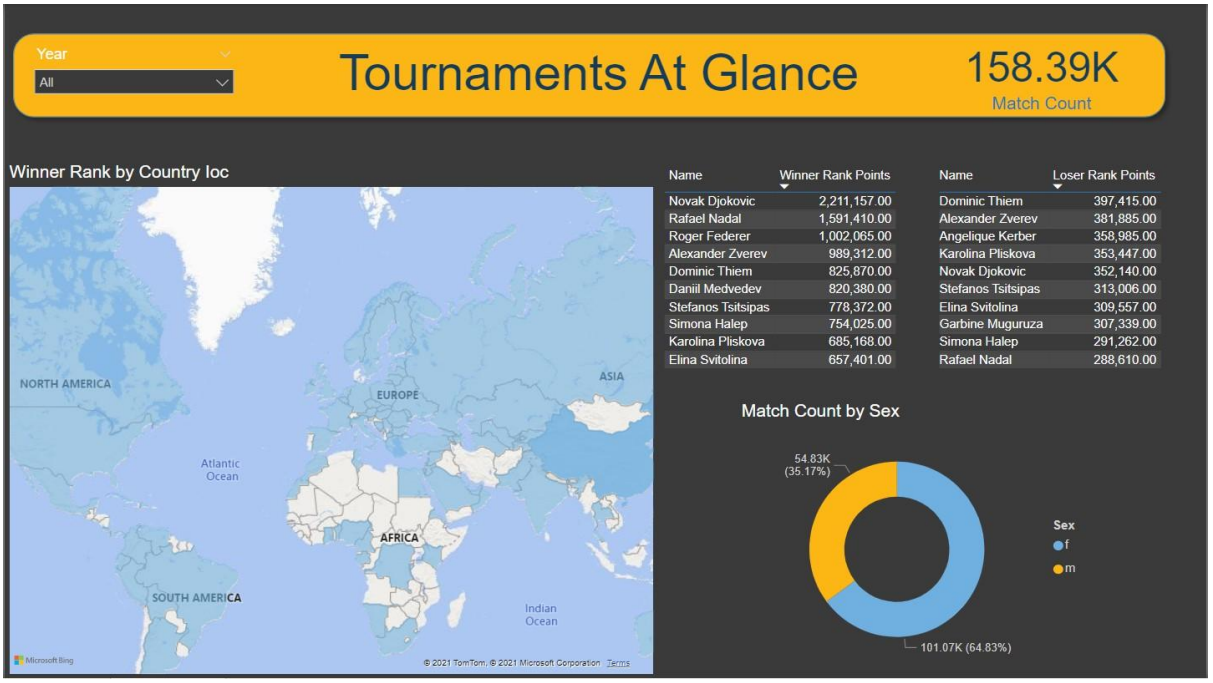
Create a dashboard that shows the geographical distribution of winner rank points and loser rank points.



In this dashboard, you have the option to drill down to see the distribution by country as well.

Assignment 5

Create a plot/dashboard of your choosing, that you deem interesting w.r.t. the data available in your cube



In the dashboard, we are showing the top10 players by the winner and loser rank point, total number of the matches as a card and distribution of players by gender and you can apply a filter on year.