UNIVERSITY OF PISA

MASTER'S DEGREE IN
DATA SCIENCE & BUSINESS INFORMATICS

LABORATORY OF DATA SCIENCE

Build a Data Warehouse

MARCO CIOMPI [537856]
LUFTJAN SALIAJ [606507]
PASQUALE GORRASI [597817]

December 30, 2021

Contents

1 Part 1

1.1 Introduction

In Part 1 of the project we were required to create and populate a database starting from .csv files and perform different operations on it. *"tennis.csv"* contains the main body of data: a fact table with tennis match data. For each match we have information about the tournament, the players involved (winner and loser) and several other metrics. Files *"male players.csv"* and *"female players.csv"* contain the list of male players and female players respectively, while *"geography.csv"* contain a list of IOC codes with country names and continents. In these four files you will find all the attributes to reproduce the schema shown in 1. The file *"tennis.csv"* will have to be split appropriately and combined with the other files to achieve this goal. The goal of the following assignments is to build the schema and deploy it on server *lds.di.unipi.it*

1.2 Assignment 0

Assignment 0 asks to recreate a Database Schema using Microsoft SQL Server Management Studio in the server *lds.di.unipi.it*

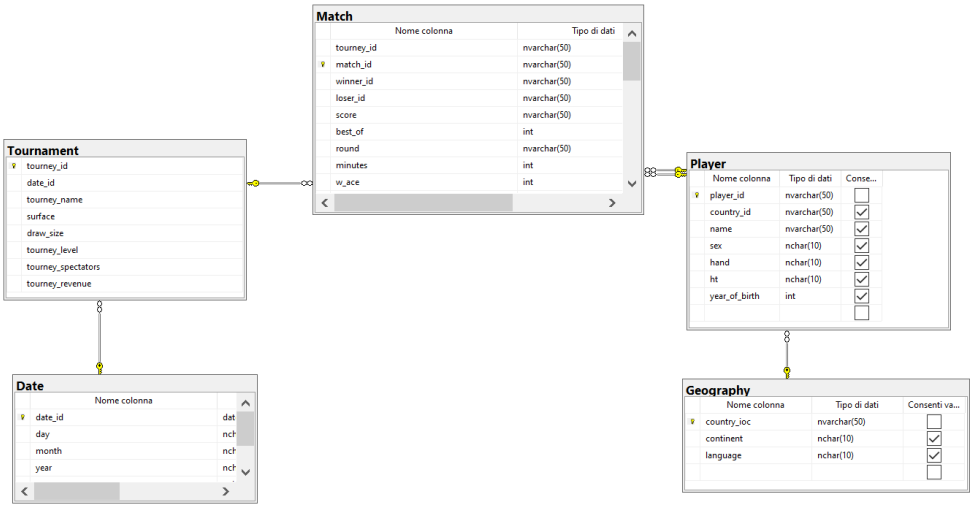

Figure 1.1: Database Schema

the keys chosen are:

- *'match id'*: for the fact table Match
- *'tourney id'*: for the table Tournament
- *'date id'*: for the table Date
- *'player id'*: for the table Player
- *'country ioc'*: for the table Geography

The foreign key relations are the followings:

- FK-Match-Tourneament: the attributes *tourney id*
- FK-Tournament-Date: the attributes *date id*
- FK-Match-Players: the attributes *winner id,loser id* on one side and *player id* on the other
- FK-Player-Geography: the attributes *country id/country ioc*

1.3 Assignment 1

1.3.1 Split Tennis.csv

Assigment 1 asks to split the given csv files to obtain the data we need to populate the database schema designed in the previous assignment, without using pandas library. To extract the five .csv files needed from *"tennis.csv"* the following code has been used, modifying for every file the column index to copy from the table.

```
columns=[12,19]

with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\countries.csv', 'w') as out:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\tennis.csv', 'r')as f:
        count = 0
        max_col = float('-inf')
        min_col = float('inf')
        for line in f:
            count +=1
            tokens = line.strip().split(',')
            new_line = ''
            for col in columns:
                new_line+=tokens[col] + ','
            new_line = new_line[:-1] + "\n"
            out.write(new_line)
            max_col = max(max_col, len(tokens))
            min_col = min(min_col, len(tokens))
        print(count, max_col, min_col)
```

Figure 1.2: write new csv files from tennis.csv

The csv files *"tournament.csv"* and *"geography.csv"* have not been modified except for the header, while for the other csv files other transformations were necessary.

Concerning the file *"date.csv"*, since the original table has just a string with the complete date, a splitting has been operated in order to obtain the attributes *"day,month,year"* and a mathematical operation to obtain the attribute *"quarter"*.

```
with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\date.csv', 'r') as f:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\datesplit.csv', 'w',newline='') as out:

        writer = writer(out)
        writer.writerow(['date_id', 'year', 'month', 'day','quarter'])
        next(f)
        reader = reader(f)

        count_id = 0
        for row in reader:

            year = int(row[0][:4])
            month = int(row[0][4:6])
            day = int(row[0][6:8])
            quarter = month//4 + 1

            row.append(year)
            row.append(month)
            row.append(day)
            row.append(quarter)

            writer.writerow(row)
```

Figure 1.3: code for splitting dates

Regarding the file *"match.csv"* we created the attribute *"match-id"* that will be used as a key for the table. We did this by adding to the attribute *"tourney-id"* a number generated by a counter that goes back to 0 when the *"tourney-id"* changes, in order to have a different value for any record.

```
with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\match.csv', 'r') as f:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\match_conid.csv', 'w',newline='') as out:

        writer = writer(out)
        writer.writerow(['tourney_id','match_id', 'match_num','winner_id','winner_age','loser_id','score','best_of','rou
        next(f)
        reader = reader(f)

        num = 0
        torneo = 0
        for row in reader:
            if row[0]!=torneo:
                num = 0
            num+=1
            torneo = row[0]

            match_id = str(num)+'-'+ torneo
            row.insert(1, match_id)

            writer.writerow(row)
```

Figure 1.4: code for generating id

Last but not least, the file *"player.csv"* required the greatest efforts. At first we created two different csv files named *winners* and *losers* using the code previously reported in *Figure 3.1.* Then we merged the two files in order to have a complete list of all the players and their features despite the final result of their games. The last step was generating a new attribute *sex* using the other csv files given to us: *female-players.csv* and *male-players.csv* that contains name and surname of all the players regarding their gender. To get the informations we needed from this tables and generating the new attribute we used the following code:

```
femalesex = open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\female_players.csv', 'r')
malesex = open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\male_players.csv', 'r')

with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\player.csv', 'r') as f:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\player_wsex.csv', 'w', newline='') as out:

        writer = writer(out)
        writer.writerow(['player_id','name','hand','ht','country_id','byear_of_birth','sex'])
        next(f)
        reader = reader(f)


        female_name = []
        male_name = []

        for row in femalesex:
            row = row.replace(',',' ').rstrip('\n')
            female_name.append(row)

        for row in malesex:
            row = row.replace(',',' ').rstrip('\n')
            male_name.append(row)

        female_name.remove(female_name[0])
        male_name.remove(male_name[0])

        female_name = set(female_name)
        male_name = set(male_name)


        for line in reader:
            if line[1] in female_name:
                line.append('female')
            elif line[1] in male_name:
                line.append('male')
            else:
                line.append('null')

            writer.writerow(line)
```

Figure 1.5: code for generating sex attribute

1.3.2 Transformation

After the splitting and formatting with python the csv files needed to populate the Data Warehouse, we cleaned them using the python library **pandas**.

The files *"geography.csv"* has just one value that presented the values "Unknown" regarding the Pacific Islands, it has been fixed. *"Date.csv"* did not show missing values, so it's been just cleaned up from duplicate values. The file *"tournament.csv"* had just a few missing values for the attribute *"surface"*, they were replaced by the mode *"Hard"*

The file *"match.csv"* presented a large number of missing values, especially for the attributes regarding the statistics. We decided to eliminate only the rows with missing values in the attributes *score, winner id, loser id*, keeping all the others.

The file *"player.csv"* was cleaned out by duplicates and presented the following missing values:

hand	32
ht	9542
byear-of-birth	2092
sex	30

For the attribute *"sex"*, since the missing values were just a few, we inferred them by looking for the player's name on Google.
For the attribute *"ht"* we decided to infer it making an average after grouping players by sex and filling the null values with the average height for their gender.
The attribute *"hand"* has been filled with his mode *U*.
Finally the missing values for the attribute *"byear-of-birth"* were left, after transforming the column that presented the age of the players instead of their birthdate.

Concerning the key relations between *geography* and *player* we verified that all the values that are in the foreign key *'country ioc'* are present also in the primary key in the table players *'country id'* comparing the intersection between the two sets of values. We discovered different values missing or typing errors and we fixed them before uploading the tables. The same was done regarding the key relation between *player id* and *winner id/loser id*.

1.4 Assignment 2

1.4.1 Uploading Data

Once fixed all the key relations between the tables we populated the tables on microsoft sql using the following python code:

```
import pyodbc
import csv

server = 'tcp:131.114.72.230'
database = 'Group_13_DB'
username = 'Group_13'
password = 'RFCS36XL'

connectionString = 'DRIVER={ODBC Driver 17 for SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password
cnxn = pyodbc.connect(connectionString)

tables = ['date','player','match','geography','tournament']

for table in tables:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\tables\\' + table + '.csv', 'r') as csv_table:
        reader = csv.reader(csv_table)
        columns = next(reader)
        query = 'insert into'+' '+ table+'({0}) values ({1})'
        query = query.format(','.join(columns), ','.join('?' * len(columns)))
        cursor = cnxn.cursor()
        for row in reader:
            cursor.execute(query, row)
        cursor.commit()

cursor.close()
```

Figure 1.6: python code for populating tables

2 Part 2

2.1 Introduction

In Part 2 of the project we are required to solve some problems on the database we created in Part 1. The exercises has to be solved using Sequel Server Integration Services (SSIS) with computation on client side

2.2 Assignment 0 - *For every tournament, the players ordered by number of matches won.*

To obtain this list we retrieved data from the tables *Match* and *Player*, respectively the *tourney id, winner id* columns and the *player id* column. Then we merge the two databases ordering the id columns, and then we aggregate in order to have for every *tourney id* the list of players and for every *player id* the number of victories, retrieved by counting the *winner id*. Finally the list has been written on a csv file.

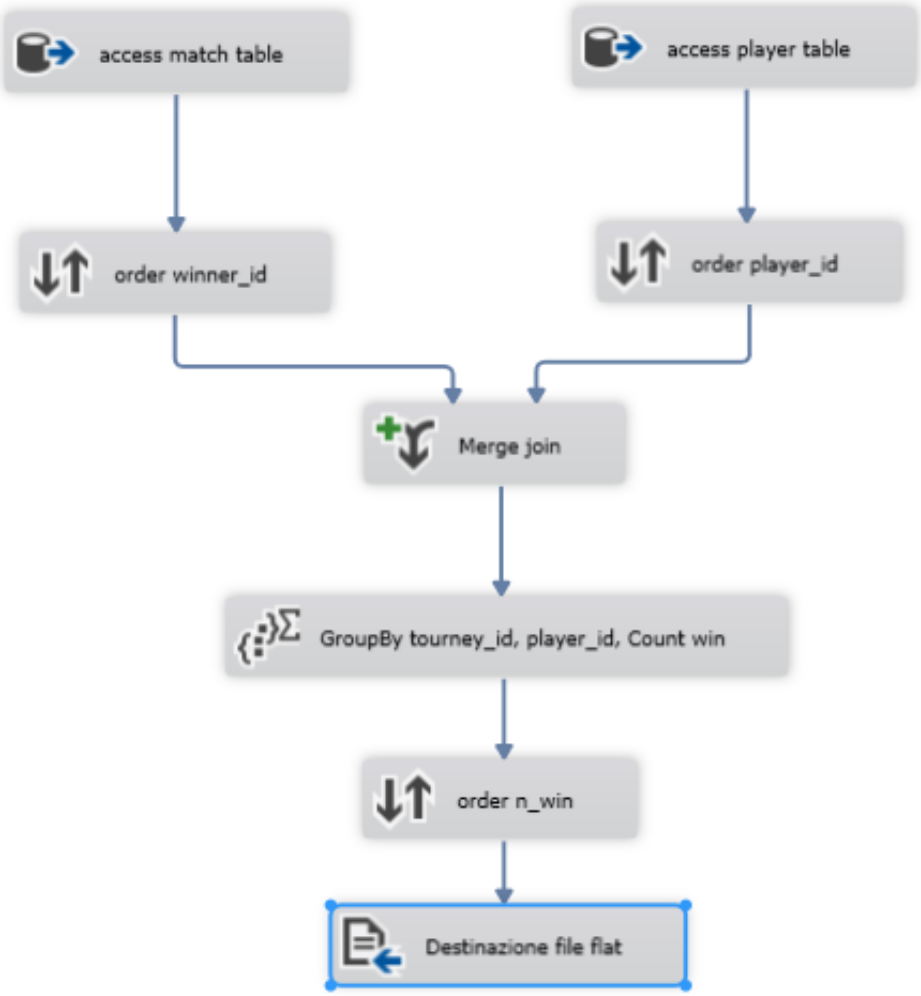

Figure 2.1: SSIS process to obtain the list.

2.3 Assignment 1 - *A tournament is said to be "worldwide" if no more than 30 percent of the participants come from the same continent. List all the worldwide tournaments.*

This assignment required an intricate procedure in order to avoid data loss and calculate the required percentage through the aggregation nodes.
The data were extracted from the table *Match*, the attributes *tourney id*, *loser id* and *winner id*. Then we appended the *country id* column from *Players* to tie the *continent* column from *Geography* using the lookup nodes. This step was done twice exploiting the multicast and union node in order not to lose data, (selecting just one between *loser id* and *winner id*), to start the process. Therefore we discarded the duplicate in *player id*.
Another multicast was required in order to aggregate by *tourney id* counting the total players on one side and all the distinct continent on the other, then we merged on *tourney id*.
Next we added a derivate column node to calculate the percentage of players coming from any continent and we kept just the higher percentage for any different tournament using an aggregation. This lead to a conditional subdivision fixing the percentage < 0.3 and the result is just one tourney considered "worldwide" by this logic.

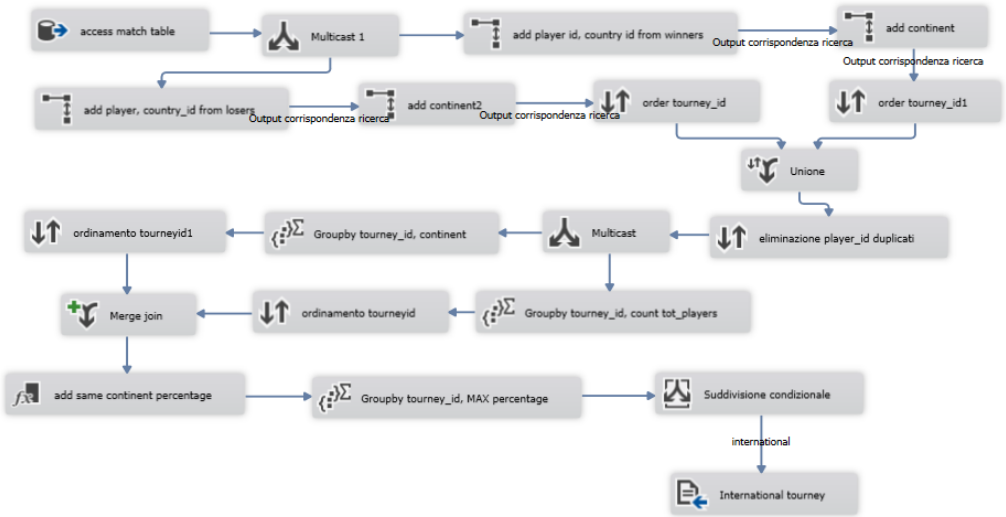

Figure 2.2: SSIS process to obtain the tournament.

2.4 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.*

For this query as in the previous one we gained data from the *Match* table casting for winners and losers and summing up the number of match won and lost by any player, merging at the end of the process. Then we retrieved *country id* from *Player* table.
Next we added a column with the derivate column node calculating the average of victories for any players, then with an aggregation node we calculated the winning average for any country. Finally with a conditional subdivision node we filtered just the players with a winning average above the average of their country.

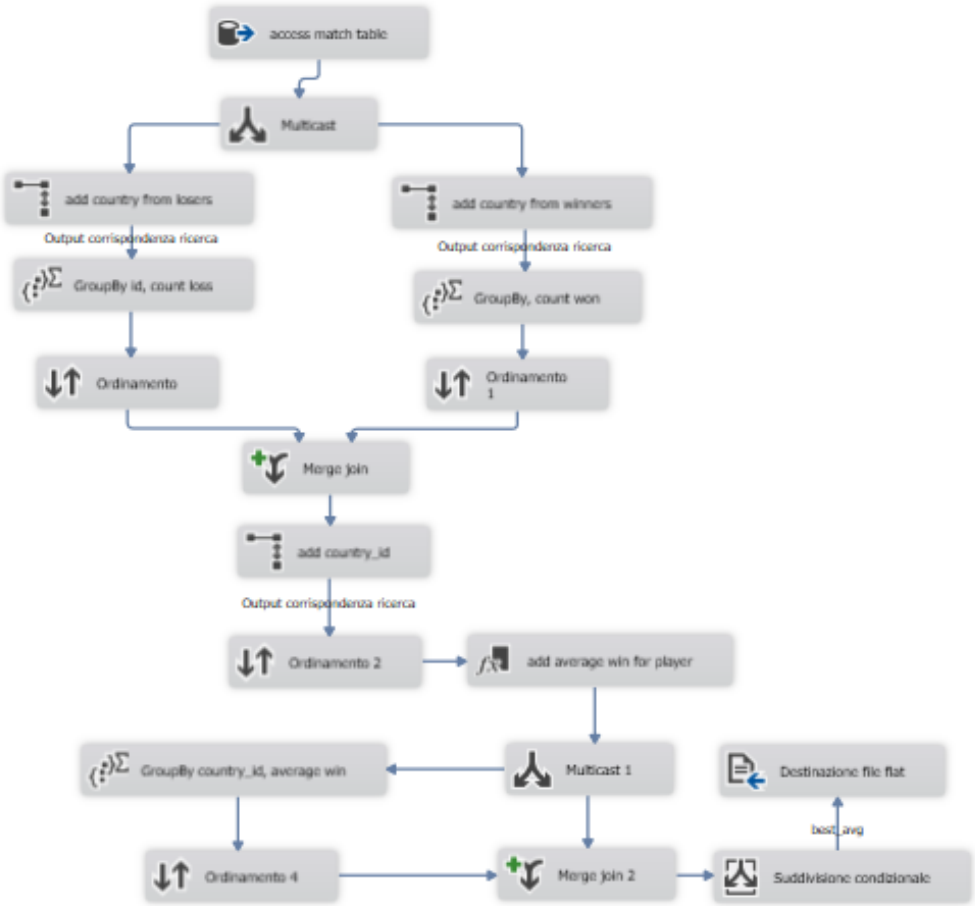

Figure 2.3: SSIS process to obtain the tournament.

3 Part 3

3.1 Introduction

In Part 3 of the project we are required to answer some business questions on a datacube that we will create on the database already prepared. Document how the datacube is being built and solve the business questions using MultiDimensional eXpressions (MDX) in SQL management studio.

3.2 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 measure.*

After establishing a connection to the datacube GROUP13CUBE with **Visual Studio**, a view of such schema was created, containing the following tables: *Match*, *Player*, *Tournament*, *Geography* and *Date*.

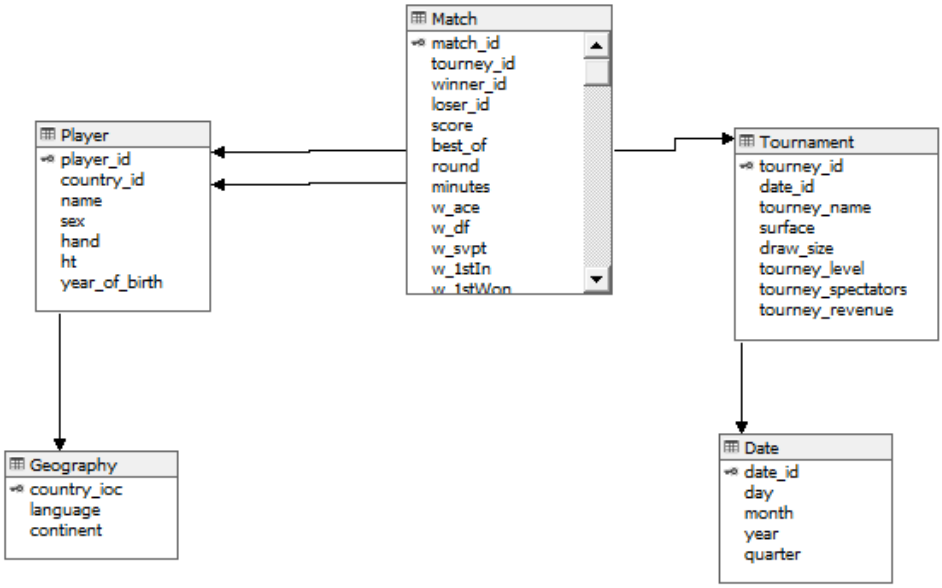

Figure 3.1: the resulting schema on Visual Studio

Starting from such view, an OLAP cube was later generated by selecting as measures *Conteggio di Match*, *Conteggio di WinnerId*, *Loser Rank*, *Loser Rank Points*, *Winner Rank*, *Winner Rank Points* from **Match** and as dimensions *Country*, *Loser*, *Winner*, *Player* and *Tournament*. Two distinct hierarchies were then built, *'YearQuarterMonthTourney'* in the Tournament dimension and *'ContCountryPlayer'* in the player dimension which was then propagated to Winner and Loser dimensions.

(a) ContCountryPlayer

(b) YearQuarterMonth-Tourney

Figure 3.2: Hierarchies

3.3 MDX queries

Once completed the designing of the cube, it was possible to solve some business questions, but this time exploiting **Sequel Server Analysis Services (SSAS)**, in particular using MultiDimensional eXpressions (MDX) in **SQL Server Management Studio**.

3.3.1 Assignment 1 - *Show the player that lost the most matches for each country.*

In order to answer this query, we created a calculated member using the MDX RANK function, which takes as input the tuple *CountryIoc.CurrentMember*, the PlayerId from the *CountCountryPlayer* Hierarchy and the set of Losers in each Country in this way we obtained the ranking among the Losers for each Countries based on the number of Matches Lost. Then we filtered the result for **rk = 1** and discarded empty results.

```
with member rk as
rank (([Loser].[Country Ioc].currentmember,[Loser].[ContCountryPlayer].currentmember),
([Loser].[Country Ioc].currentmember, [Loser].[ContCountryPlayer].[Player Id]),
[Measures].[Conteggio di Match])
select [Measures].[Conteggio di Match] on columns,
nonempty(filter(([Loser].[Country Ioc].[Country Ioc], [Loser].[ContCountryPlayer].[Player Id]), rk = 1)) on rows
from [Group 13 DB]
```

Figure 3.3: MDX query assignment 1

		Conteggio di Match
AHO	Laurens Deelstra	1
ALG	Ines Ibbou	68
AND	Victoria Jimenez Kasintseva	22
ARG	Renzo Olivo	156
ARM	Ani Amiraghyan	34
AUS	Jordan Thompson	141
AUT	Sebastian Ofner	132
AZE	Amine Dik	3
AZE	Fidana Khalilzada	3
BAH	Kerrie Cartwright	13
BAN	Jhilik Chakma	1
BAR	Darian King	101
BDI	Sada Nahimana	30
BEL	Kimmer Coppejans	145
BEN	Alexis Klegou	1

Figure 3.4: MDX query head result assignment 1

3.3.2 Assignment 2 - *For each tournament, show the loser with the lowest total loser rank points.*

For the purpose of answering such query, We used the MDX GENERATE function to apply a specific operation (MDX BOTTOMCOUNT) to every member of given set, joining the resulting sets by union. By doing so, the Loser player with the lowest rank points (BOTTOMCOUNT sorts the set of losers for each TournmentYear in ascending order and returns the first tuple which is the one with the lowest value) was individuated for each Tourney Id and then reported with the following schema: the pair *TourneyIdYear - PlayerId* on the rows, *LoserRankPoints* on the columns.

```
SELECT [Measures].[Loser Rank Points] ON COLUMNS,
GENERATE(([Tournament].[Tourney Id].[Tourney Id], [Tournament].[Year].[Year]),
BOTTOMCOUNT(([Tournament].[Tourney Id].CURRENTMEMBER,[Tournament].[Year].currentmember,
nonempty(FILTER([Loser].[Player Id].[Player Id], [Measures].[Loser Rank Points] > 0))),
1,
[Measures].[Loser Rank Points])) ON ROWS
FROM [Group 13 DB]
```

Figure 3.5: MDX query assignment 2

			Loser Rank Points
Abu Dhabi	2021	Makoto Ninomiya	20
Acapulco	2016	Dmitry Tursunov	10
Acapulco	2017	Manuel Sanchez	38
Acapulco	2018	Alan Fernando Rubio Fierros	1
Acapulco	2019	Luis Patino	6
Acapulco	2020	Lucas Gomez	10
Acapulco	2021	Luis Patino	15
Adelaide	2020	Mikalai Haliak	10
Adelaide	2021	Kimberly Birrell	41
Agadir $15K	2017	Linda Puppendahl	3
Aix en Provence CH	2016	Lofo Ramiaramanana	1
Aix En Provence CH	2017	Mohamed Nazim Makhlouf	2
Aix En Provence CH	2018	Hugo Gaston	6
Aix En Provence CH	2019	Hugo Gaston	4
Aix En Provence CH	2020	Giovanni Mpetshi Perricard	2
Aix-En-Provence CH	2021	Titouan Droguet	34
Akko $10K	2016	Christina Shakovets	3
Akko $15K	2017	Nicole Nadel	4
Akko $15K	2018	Taysia Rogers	3
Albuquerque NM $75K	2016	Chanelle Van Nguyen	18
Albuquerque NM $80K	2017	Safiya Carrington	3

Figure 3.6: MDX query head result assignment 2

3.3.3 Assignment 3 - *For each tournament, show the loser with the highest ratio between his loser rank points and the average winner rank points of that tournament.*

Aiming to solve this last query, we created three different CALCULATED MEMBERS in order to obtain the ratio of each Player. The first member that we created was *Media* which is the average of winner rank points of players in each tournment. The next calculated member was *LoserRank* in order to obtain the Loser Rank Points of each player for each tournment. After that we could properly calculate the ratio between Loser Rank Points and the average winner rank points of a specific tournament for each Loser Players which was stored in the third calculated member *Ratio*. Moreover, we used the MDX GENERATE function in order to apply a specific operation (MDX TOPCOUNT) to every member of given set, joining the resulting sets by union. By doing so, Loser Players with the highest Ratio for each Tournment were spotted and then reported with the following schema: the pair *Tourney-IdYear - PlayerId* on the rows, *Media*, *LoserRank* and *Ratio* on the columns.

```
with member media as
aggregate(([Tournament].[Tourney Id].CURRENTMEMBER, [Loser].[Player Id].[All]),
[Measures].[Winner Rank Points])/
aggregate(([Tournament].[Tourney Id].CURRENTMEMBER, [Loser].[Player Id].[All]),
[Measures].[Conteggio di Match])
member loserrank as
([Tournament].[Tourney Id].CURRENTMEMBER, [Measures].[Loser Rank Points])
member ratio as
case when media = 0 then 0 else loserrank / media end
select {media, loserrank, ratio}  on columns,
GENERATE(([Tournament].[Tourney Id].[Tourney Id], [Tournament].[Year].[Year]),
TOPCOUNT(([Tournament].[Tourney Id].CURRENTMEMBER, [Tournament].[Year].CURRENTMEMBER,
nonempty(FILTER([Loser].[Player Id].[Player Id], [Measures].[Loser Rank Points] > 0))),
1, ratio)) on rows
from [Group 13 DB]
```

Figure 3.7: MDX query assignment 3

			media	loserrank	ratio
Abu Dhabi	2021	Sofia Kenin	1567.1724137931	5760	3.67540925893329
Acapulco	2016	Kei Nishikori	1013.39130434783	4235	4.17903724043247
Acapulco	2017	Novak Djokovic	1355.40217391304	9735	7.18237006503765
Acapulco	2018	Alexander Zverev	1224.11956521739	4450	3.63526580772339
Acapulco	2019	Rafael Nadal	1227.5	8320	6.77800407331976
Acapulco	2020	Alexander Zverev	1234.69565217391	3885	3.14652440312698
Acapulco	2021	Stefanos Tsitsipas	1702.96610169492	6765	3.97248071659617
Adelaide	2020	Simona Halep	1581.87058823529	5461	3.45224194735942
Adelaide	2021	Ashleigh Barty	1515.74418604651	9186	6.06038940116912
Agadir $15K	2017	Pia Konig	26.8709677419355	46	1.71188475390156
Aix en Provence CH	2016	Lukas Rosol	267.102040816327	737	2.75924511002445
Aix En Provence CH	2017	Malek Jaziri	315.592592592593	726	2.30043422133552
Aix En Provence CH	2018	Jeremy Chardy	246.642857142857	745	3.0205618302925
Aix En Provence CH	2019	Lloyd Harris	300.387755102041	631	2.10061824852232
Aix En Provence CH	2020	Gilles Simon	334.46511627907	970	2.90015296898901
Aix-En-Provence CH	2021	Facundo Bagnis	293.139534883721	828	2.82459341531138
Akko $10K	2016	Naomi Totka	32.6129032258064	72	2.20771513353116
Akko $15K	2017	Ana Veselinovic	25.3548387096774	80	3.15521628498728
Akko $15K	2018	Caroline Uebelhoer	13	49	3.76923076923077
Albuquerque NM $75K	2016	Alison Van Uytvanck	361.741935483871	625	1.72775102550383
Albuquerque NM $80K	2017	Viktorija Golubic	82.8	560	6.76328502415459

Figure 3.8: MDX query head result assignment 3

3.4 Dashboards

The last step of the process described in this report was to create some dashboards in order to give the user a deeper insight on some dimensions of the data. The software exploited for creating these panels were **Power BI**.

3.4.1 Assignment 4 - *Create a dashboard that shows the geographical distribution of winner rank points and loser rank points.*

The first goal was to create a dashboard showing the geographical distribution of Winner Rank Points and Loser Rank Points. In order to do so two grouped bars chart and a line graph were created, highlighting the Loser Rank Points and the Winner Rank Points for Country Ioc. During the design phase, we were willing to exploit a geographical map in order to show the distribution, but deepening the analysis we found out that some Country Ioc were not in line with the Power BI mapping, thus creating misleading representations for some particular countries. For this reason we chose different graphical representations, clearly showing the countries that has the highest ranking points. The dashboard is reported in a dynamic environment, thus creating filtering options among the different graphs.

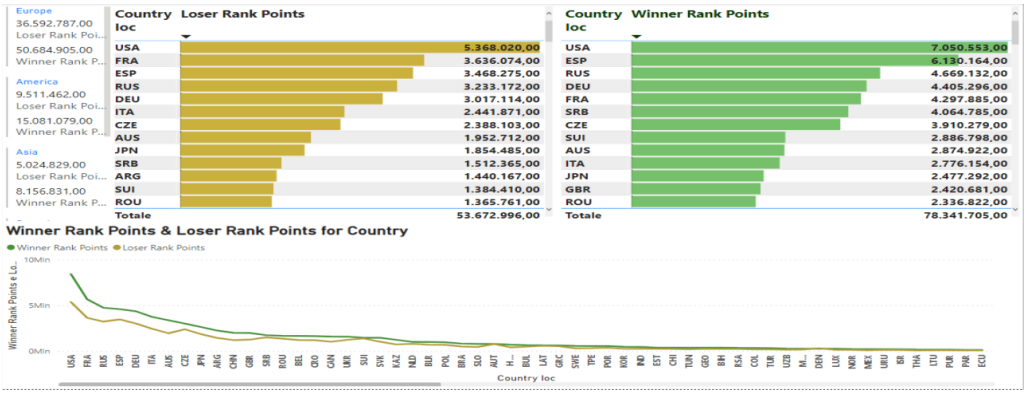

Figure 3.9: Power BI Dashboard 1

3.4.2 Assignment 5 - *Create a plot/dashboard of your choosing, that you deem interesting w.r.t. the data available in your cube*

This last dashboard shows the number of winners for continent and sex in three different ways. An interactive geographical distribution showing a pie chart for each continent where you can see the number of winners divided by sex and clicking on it we can isolate only the sex of interest to better visualize it on the scatter chart and on the histogram. In the developed scatter chart we exploited the *YearQuarterMonthPlayer* hierarchy, giving the opportunity of drill-down operations to highlight the number of winners linked to the continent where they are located, therefore a linear relationship. Through this chart it is clearly visible the European dominance of winners over time. The histogram shows the descending distribution of winners for country and sex. The described dashboard is shown in the Figure 3.2

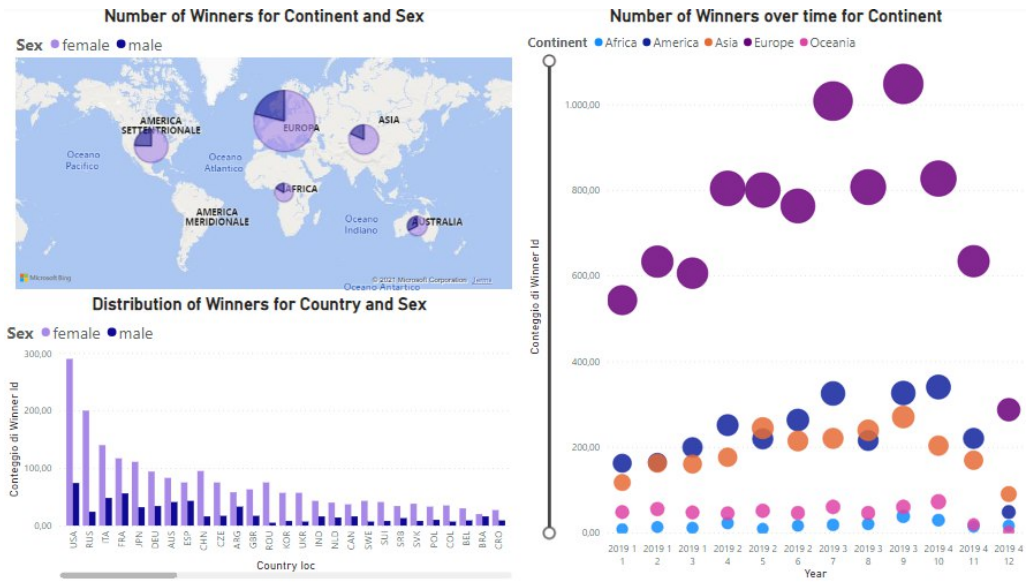

Figure 3.10: Power BI Dashboard 2